# High-Speed Polynomial Basis Multipliers Over $GF(2^m)$ for Special Pentanomials

José L. Imaña

***Abstract*—Efficient hardware implementations of arithmetic operations in the Galois field $GF(2^m)$ are highly desirable for several applications, such as coding theory, computer algebra and cryptography. Among these operations, multiplication is of special interest because it is considered the most important building block. Therefore, high-speed algorithms and hardware architectures for computing multiplication are highly required. In this paper, bit-parallel polynomial basis multipliers over the binary field $GF(2^m)$ generated using type II irreducible pentanomials are considered. The multiplier here presented has the lowest time complexity known to date for similar multipliers based on this type of irreducible pentanomials.**

***Index Terms*—Bit-parallel multipliers, finite field, $GF(2^m)$, irreducible pentanomials, polynomial basis.**

## I. Introduction

BINARY GALOIS field arithmetic is a widely studied subject due to its use in several important applications. $GF(2^m)$ arithmetic only requires AND and XOR gates for its implementation. XOR-based logic functions have been studied since the 1960s [1] due to their use in coding theory [2], digital signal processing, cryptography and telecommunication circuits. These applications frequently require efficient very large scale integration (VLSI) implementations of high speed $GF(2^m)$ multipliers [3]–[9]. For this reason, several bit-parallel *polynomial basis* (PB) multipliers have been proposed. Polynomial basis is the most widely used, although *normal* [10] or *dual* [11] basis can also be considered. The complexity of the multiplier depends on the generating irreducible polynomial $f(y)$ selected for the finite field. For hardware implementation of $GF(2^m)$ multiplication, low Hamming weight irreducible polynomials, such as trinomials and pentanomials, are usually used. For irreducible trinomials, multipliers with low area and time complexities can be implemented [12]–[14]. Unfortunately, there are 468 values of $m$ in the interval [2,1024] such that irreducible trinomials of degree $m$ do not exist. For each of the other values of $m$ in the same range, where no such irreducible trinomial exist, an irreducible pentanomial can be found. Thus, the design of multipliers using irreducible pentanomials is needed. Polynomial basis multiplication requires a polynomial multiplication followed by a modular reduction. An efficient bit-parallel multiplier was proposed by Mastrovito [15] in which a *product matrix* is introduced to combine the above two steps together. The entries in this matrix can be computed efficiently by sharing common items, known as *subexpression sharing* [16]. Mastrovito multipliers using special irreducible pentanomials have been widely studied due to their low-complexity implementations [12], [13], [17], [18]. All these works exploit subexpression sharing in order to find efficient architectures. Other methods use the divide-and-conquer approach for polynomial multiplication in order to reduce the complexity of the multiplier [19], [21]. In [9], a new PB multiplication method was used. This method is based on the introduction of a product matrix that can be decomposed as a sum of matrices depending on the selected irreducible polynomial. Matrix *decomposition* was already used in similar $GF(2^m)$ multiplication approaches that exploit subexpression sharing [12], [13], [17], [18]. The method in [9] introduced the functions $\mathbf{S_i}$ and $\mathbf{T_i}$ given by the raw sum of terms $x_k = (a_k b_k)$ and $z_{i,j} = (a_i b_j + a_j b_i)$, where $a_i, b_i \in GF(2)$ are the coefficients of two $GF(2^m)$ elements $A$ and $B$, respectively. The coefficients of the product of two field elements can be computed as the sum of that functions. One of the problems of the above method is related with the monolithic construction of the $\mathbf{S_i}$ and $\mathbf{T_i}$ functions. For example, for $GF(2^6)$ the functions $\mathbf{T_2} = x_4 + z_{3,5} = a_4 b_4 + (a_3 b_5 + a_5 b_3)$ and $\mathbf{T_4} = x_5 = a_5 b_5$ are defined. The sum of these two functions $\mathbf{T_2} + \mathbf{T_4} = ((a_4 b_4 + (a_3 b_5 + a_5 b_3)) + a_5 b_5)$ would result in a 3-level (with depth 3) binary tree of XOR gates. However, the sum of $\mathbf{T_2} + \mathbf{T_4}$ involves the addition of four product terms ($a_4 b_4$, $a_3 b_5$, $a_5 b_3$ and $a_5 b_5$) and it could be done with a 2-level complete binary tree of XOR gates.

In this work, a new bit-parallel PB multiplier is presented by considering the functions $\mathbf{S_i}$ and $\mathbf{T_i}$ as a sum of $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ terms, respectively, in such a way that $\mathbf{S_i} = s_\rho^i \mathbf{S_i^\rho} + \ldots + s_1^i \mathbf{S_i^1} + s_0^i \mathbf{S_i^0}$ and $\mathbf{T_i} = t_\rho^i \mathbf{T_i^\rho} + \ldots + t_1^i \mathbf{T_i^1} + t_0^i \mathbf{T_i^0}$ for a given finite field $GF(2^m)$, where $s_j^i, t_j^i \in GF(2)$ and $\rho = \lfloor log_2 m \rfloor$. The terms $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ represent the addition of $2^j$ products $a_k b_l$ and therefore can be implemented as a $j$-level complete binary tree of XOR gates. In this way, the addition of terms $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ with the same superscript $j$ would result in a $j+1$-level complete binary tree. If the sum of the functions $\mathbf{S_i}$ and $\mathbf{T_i}$ is performed by grouping the additions of terms with the same $j$-level $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$, then the number of XOR levels needed to compute the product coefficients can be reduced. Furthermore, the coefficients $(s_\rho^i, \ldots, s_1^i, s_0^i)$ and $(t_\rho^i, \ldots, t_1^i, t_0^i)$ are given by the binary representations of the subindex $i$ for $\mathbf{S_i}$ and of the value $m-1-i$ for $\mathbf{T_i}$, respectively. In this contribution, the new multiplication approach is applied to *type II irreducible pentanomials* [18] $f(y) = y^m + y^{n+2} + y^{n+1} + y^n + 1$, with $1 \leq n \leq \lfloor m/2 \rfloor - 1$. These pentanomials are important because they are abundant (there are 597 values of $m$ in the interval

Manuscript received August 05, 2015; revised October 06, 2015; accepted October 31, 2015. This work was supported by the Spanish Government under Research Grants CICYT TIN2008-00508 and TIN2012-32180. This paper was recommended by Associate Editor S. Ghosh.

The author is with the Department of Computer Architecture and Systems Engineering, Faculty of Physics, Complutense University, 28040 Madrid, Spain (e-mail: jluimana@ucm.es).

Digital Object Identifier 10.1109/TCSI.2015.2500419

[5,1000] such that these type of irreducible pentanomials of degree $m$ exists) and because all five binary fields recommended by NIST for ECDSA, i.e., $m \in \{163, 233, 283, 409, 571\}$, can be constructed using such irreducible polynomials.

The paper is organized as follows. Notation and mathematical background are presented in Section II, where PB multiplication for type II irreducible pentanomials given in [9] is also reviewed. The new multiplication approach is presented in Section III, where an example of multiplication and the complexity analysis are also given. In Section IV comparisons with other similar multipliers are done. Finally, concluding remarks are made in Section V.

## II. Notation and Preliminaries

Let $f(y) = \sum_{i=0}^{m} f_i y^i$ be an irreducible polynomial of degree $m$ over $GF(2)$. All elements of the binary finite field $GF(2^m) = GF(2)[y]/(f(y))$ can be represented in the *polynomial basis* $\{1, x, \ldots, x^{m-1}\}$, where $x$ is a root of the polynomial $f(y)$. For example, an element $A \in GF(2^m)$ is represented in PB as $A = \sum_{i=0}^{m-1} a_i x^i = (1, x, \ldots, x^{m-1}) \cdot (a_0, a_1, \cdots, a_{m-1})^T$, with $a_i \in GF(2)$. The vector of the coefficients of $A$ in PB can be represented by $\underline{A} = (a_0, \ldots, a_{m-1})^T$. Let $A, B, C \in GF(2^m)$ and $\underline{A}, \underline{B}, \underline{C}$ be their coefficient vectors, respectively. Using the method given in [9], the product $C = A \cdot B$ can be computed as $\underline{C}^R = (c_{m-1}, \ldots, c_0) = \underline{A}^T \cdot \mathbf{K}$, where $\underline{C}^R$ is the vector of reversed coefficients of $\underline{C}$ and $\mathbf{K}$ is the product or Mastrovito matrix that depends on $f(y)$ and on the coefficients $b_i$ of $B$. In order to compute the $\underline{C}$ coefficients, a new notation was given in [9]. These coefficients consist of *sum-of-products* (SOP) given by the inner products of $\underline{A}$ and $\underline{B}$. An *inner product* can be represented by the *permutation* given by the subscripts of the coefficients of $A$ and $B$, respectively, in the SOP. From this permutation, 1-cycles $(k)$ and 2-cycles $(i, j)$, can be found and associated with the terms $x_k = (a_k b_k)$ and $z_{i,j} = (a_i b_j + a_j b_i)$, respectively. For example, the SOP $a_0 b_4 + a_1 b_3 + a_2 b_2 + a_3 b_1 + a_4 b_0$ can be represented by the cycles (0,4)(1,3)(2). In [9], the sum of the $x_k$ and $z_{i,j}$ terms represented by the 1-cycles $(k)$ and 2-cycles $(i, j)$ were carried out by the functions $\mathbf{S_i}$ and $\mathbf{T_i}$. These functions are implemented as *binary trees* of 2-input XOR gates with a lower level of 2-input AND gates (corresponding to the $a_i b_j$ products of the coefficients of $A$ and $B$). The product $C = A \cdot B$ can be computed as the sum of these functions. The expression for $\mathbf{S_i}$ $(1 \leq i \leq m)$ with $\varsigma = \lfloor i/2 \rfloor$, is [9]:

$$\mathbf{S_i} = x_\varsigma + \sum_{h=0}^{\varsigma-1} z_{h,i-h-1}, \tag{1}$$

where $x_\varsigma = a_\varsigma b_\varsigma$ only appears for $i$ *odd*. The expression for $\mathbf{T_i}$ $(0 \leq i \leq m-2)$ with $\gamma = (\lceil m/2 \rceil + \lfloor i/2 \rfloor)$ is as follows:

$$\mathbf{T_i} = x_\gamma + \sum_{j=1}^{\eta-(i+1)} z_{i+j,m-j}, \tag{2}$$

where the term $x_\gamma$ only appears for ($m$ and $i$ *even*) or for ($m$ and $i$ *odd*). In this case, $\eta = \gamma$. Otherwise, i.e., for ($m$ *even* and $i$ *odd*) or for ($m$ *odd* and $i$ *even*), the term $x_\gamma$ does not appear and the value of $\eta = (\lceil m/2 \rceil + \lceil i/2 \rceil)$. For example, for $GF(2^6)$ the terms $\mathbf{S_i}$ and $\mathbf{T_i}$ are as follows: $\mathbf{S_1} = x_0 = a_0 b_0$, $\mathbf{S_2} = z_{0,1} = (a_0 b_1 + a_1 b_0)$, $\mathbf{S_3} = x_1 + z_{0,2} = a_1 b_1 + (a_0 b_2 + a_2 b_0)$, $\mathbf{S_4} = (a_0 b_3 + a_3 b_0) + (a_1 b_2 + a_2 b_1)$, $\mathbf{S_5} = a_2 b_2 + (a_0 b_4 + a_4 b_0) + (a_1 b_3 + a_3 b_1)$, $\mathbf{S_6} = (a_0 b_5 + a_5 b_0) + (a_1 b_4 + a_4 b_1) + (a_2 b_3 + a_3 b_2)$, $\mathbf{T_0} = a_3 b_3 + (a_1 b_5 + a_5 b_1) + (a_2 b_4 + a_4 b_2)$, $\mathbf{T_1} = (a_2 b_5 + a_5 b_2) + (a_3 b_4 + a_4 b_3)$, $\mathbf{T_2} = a_4 b_4 + (a_3 b_5 + a_5 b_3)$, $\mathbf{T_3} = (a_4 b_5 + a_5 b_4)$, $\mathbf{T_4} = a_5 b_5$.

TABLE I
Expressions for the Coefficients $c_i$ of the Product

| | | |
|---|---|---|
| $c_0$ | $= S_1 + T_0 + T_{z-2} + T_{z-1} + T_z;$ | |
| $c_1$ | $= S_2 + T_1 + T_{z-1} + T_z + T_{z+1};$ | |
| ⋮ | ⋮ | Ⓐ |
| $c_{n-2}$ | $= S_{n-1} + T_{n-2} + T_{m-4} + T_{m-3} + T_{m-2};$ | |
| $c_{n-1}$ | $= S_n + T_{n-1} + T_{m-3} + T_{m-2};$ | |
| $c_n$ | $= S_{n+1} + T_n + T_{m-2} + T_0 + T_{z-2} + T_{z-1} + T_z;$ | Ⓑ |
| $c_{n+1}$ | $= S_{n+2} + T_{n+1} + T_0 + T_{z-2} + T_1 + T_{z+1};$ | |
| $c_{n+2}$ | $= S_{n+3} + T_{n+2} + T_0 + T_{z-2} + T_z + T_1 + T_2 + T_{z+2};$ | |
| $c_{n+3}$ | $= S_{n+4} + T_{n+3} + T_1 + T_{z-1} + T_{z+1} + T_2 + T_3 + T_{z+3};$ | |
| ⋮ | ⋮ | |
| $c_i$ | $= S_{i+1} + T_i + T_{i-(n+2)} + T_{z+[(i-(n+2))-2]} + T_{z+[i-(n+2)]} + T_{i-(n+1)} + T_{i-n} + T_{z+[i-n]};$ | Ⓒ |
| ⋮ | ⋮ | |
| $c_{2n-3}$ | $= S_{2n-2} + T_{2n-3} + T_{n-5} + T_{m-7} + T_{m-5} + (T_{n-4} + T_{n-3}) + T_{m-3};$ | |
| $c_{2n-2}$ | $= S_{2n-1} + T_{2n-2} + T_{n-4} + T_{m-6} + T_{m-4} + T_{n-3} + T_{n-2} + T_{m-2};$ | |
| $c_{2n-1}$ | $= S_{2n} + T_{2n-1} + T_{n-3} + T_{m-5} + T_{m-3} + T_{n-2} + T_{n-1};$ | Ⓓ |
| $c_{2n}$ | $= S_{2n+1} + T_{2n} + T_{n-2} + T_{m-4} + T_{m-2} + T_{n-1} + T_n;$ | |
| $c_{2n+1}$ | $= S_{2n+2} + T_{2n+1} + T_{n-1} + T_{m-3} + T_n + T_{n+1};$ | Ⓔ |
| $c_{2n+2}$ | $= S_{2n+3} + T_{2n+2} + T_n + T_{m-2} + T_{n+1} + T_{n+2};$ | |
| $c_{2n+3}$ | $= S_{2n+4} + T_{2n+3} + T_{n+1} + T_{n+2} + T_{n+3};$ | |
| ⋮ | ⋮ | Ⓕ |
| $c_{m-2}$ | $= S_{m-1} + T_{m-2} + T_{z-4} + T_{z-3} + T_{z-2};$ | |
| $c_{m-1}$ | $= S_m + T_{z-3} + T_{z-2} + T_{z-1};$ | Ⓖ |

*Type II irreducible pentanomials* were defined in [18] as $f(y) = y^m + y^{n+2} + y^{n+1} + y^n + 1$, for $1 \leq n \leq \lfloor m/2 \rfloor - 1$. In this expression, if $n = 1$ then a *type I* pentanomial [20] is obtained. Polynomial basis multiplication for type II irreducible pentanomials with $n \geq 2$ was studied in [9]. The coefficients of the PB product were given as the sum of $\mathbf{S_i}$ and $\mathbf{T_i}$ terms, as shown in Table I, where $z = m - n$. In this table, the coefficients have been divided into seven sections (named from Ⓐ to Ⓖ), depending on the number of $\mathbf{S_i}$ and $\mathbf{T_i}$ terms in the sums. The first section Ⓐ(from $c_0$ to $c_{n-2}$) has 5 terms; section Ⓑ with $c_{n-1}$, $c_n$ and $c_{n+1}$ has 4, 7 and 6 terms, respectively; section Ⓒ($c_{n+2}$ to $c_{2n-2}$) has 8 terms; sections Ⓓ($c_{2n-1}$, $c_{2n}$) and Ⓔ($c_{2n+1}$, $c_{2n+2}$) have 7 and 6 terms, respectively; section Ⓕ($c_{2n+3}$ to $c_{m-2}$) has 5 terms; and section Ⓖ ($c_{m-1}$) has 4 terms. From (2), it can be observed that the term $\mathbf{T_0}$ is given by the addition of $\eta - 1$ terms $z_{i,j}$ and the term $x_\gamma$ (if it exists). Therefore, $\mathbf{T_0}$ performs the sum of the maximum number of $z_{i,j}$ terms among $\mathbf{T_i}$ ones and it presents the highest delay. As this term appears in the coefficient $c_{n+2}$ that is included in section Ⓒ with the maximum number of terms, then $c_{n+2}$ is the coefficient with the highest delay of the multiplier. In the following section, a new scheme for multiplication is given.

## III. New Multiplier for Type II Irreducible Pentanomials

The functions $\mathbf{S_i}$ and $\mathbf{T_i}$ presented in (1), (2) are given by a raw sum of terms $x_k = (a_k b_k)$ and $z_{i,j} = (a_i b_j + a_j b_i)$. The coefficients of the product of two field elements represented in PB can be computed as the sum of that functions, as given in Table I. One of the problems of the above method is related

with the monolithic construction of the $\mathbf{S_i}$ and $\mathbf{T_i}$ functions. For example, for $GF(2^6)$ the functions $\mathbf{T_2} = x_4 + z_{3,5} = (a_4b_4 + (a_3b_5 + a_5b_3))$ and $\mathbf{T_4} = x_5 = a_5b_5$ are defined. The sum of these two functions $\mathbf{T_2} + \mathbf{T_4} = ((a_4b_4 + (a_3b_5 + a_5b_3)) + a_5b_5)$, where the terms in brackets point out that they must be added (XOR) previously to the XOR with the other terms, would result in a 3-level (with depth 3) binary tree of XOR gates. However, the sum of $\mathbf{T_2}+\mathbf{T_4}$ involves the addition of four product terms ($a_4b_4$, $a_3b_5$, $a_5b_3$ and $a_5b_5$) so it could be done with a 2-level complete binary tree of XOR gates if the involved additions could be performed in a separate way, i.e., if the product $a_4b_4$ could be first added with the term $a_5b_5$ and then perform the addition with $(a_3b_5 + a_5b_3)$ in the form $\mathbf{T_2} + \mathbf{T_4} = ((a_4b_4 + a_5b_5) + (a_3b_5 + a_5b_3))$.

**Algorithm 1** Computation of initial $\mathbf{S_i^j}$ terms of $\mathbf{S_i}$.

**for** $i = 1$ to $m$ **do**
  **if** odd $i$ **then**
    $\mathbf{S_i^0} = x_{\lfloor i/2 \rfloor}$;
  **else**
    $\mathbf{S_i^0} = \emptyset$;
  **end if**
  $l = 0$;
  **for** $k = 1$ to $\lfloor log_2 m \rfloor$ **do**
    **if** $\lfloor i/2^k \rfloor \; mod \; 2 = 1$ $\{\exists \; \mathbf{S_i^k}$ at level $k\}$ **then**
      $\mathbf{S_i^k} = \sum_{h=l}^{\pi} z_{h,i-h-1}$; $l = \pi + 1$;
    **else**
      $\mathbf{S_i^k} = \emptyset$;
    **end if**
  **end for**
**end for**

In this paper, a new bit-parallel PB multiplier is presented by considering the functions $\mathbf{S_i}$ and $\mathbf{T_i}$ as a sum of $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ terms, respectively, in such a way that $\mathbf{S_i} = s_\rho^i \mathbf{S_i^\rho} + \ldots + s_1^i \mathbf{S_i^1} + s_0^i \mathbf{S_i^0}$ and $\mathbf{T_i} = t_\rho^i \mathbf{T_i^\rho} + \ldots + t_1^i \mathbf{T_i^1} + t_0^i \mathbf{T_i^0}$ for a given finite field $GF(2^m)$, where $s_j^i, t_j^i \in GF(2)$ and $\rho = \lfloor log_2 m \rfloor$. The initial terms $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ represent the addition of $2^j$ products $a_k b_l$ and therefore can be implemented as a $j$-level complete binary tree of XOR gates. In this way, the addition of two terms $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ with the same superscript $j$ would result in a new XOR in the level $j+1$ (i.e., a new $j+1$-level term) that represents a $j+1$-level complete binary tree. If the sum of the functions $\mathbf{S_i}$ and $\mathbf{T_i}$ is performed by grouping the additions of terms with the same $j$-level $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$, starting with the lower levels, then the number of XOR levels needed to compute the product coefficients can be reduced. In this way, the 0-level initial terms $\mathbf{S_i^0}$ and $\mathbf{T_i^0}$ should be first added in pairs to give rise to a new XOR in the level 1 (i.e., a new 1-level binary tree term), that in turn should be added with other 1-level term to give rise to a new 2-level complete binary tree and so on. If there is only one $j$-level term (or there is an unpaired $j$-level term), then it should be added with an immediately above $(j+1)$-level term in order to have a new $(j+2)$-level tree. If no such a $(j+1)$-level term exists, then it should be added with a $(j+2)$-level term, and so on.

From (1), (2), the computations of the initial terms $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ of $\mathbf{S_i}$ and $\mathbf{T_i}$ are given in Algorithm 1 and Algorithm 2, respectively, where the term $\pi = l + (2^{k-1} - 1)$ has been used. In these algorithms, the condition $\lfloor i/2^k \rfloor \; mod \; 2 = 1$ in the inner **for** loop determines if the $\mathbf{S_i}$ or $\mathbf{T_i}$ terms have an initial term $\mathbf{S_i^j}$ or $\mathbf{T_i^j}$ at level $k$. This condition will be further explained in Section III-B.

**Algorithm 2** Computation of initial $\mathbf{T_i^j}$ terms of $\mathbf{T_i}$.

**for** $i = 0$ to $m - 2$ **do**
  **if** (even $m$ and $i$) or (odd $m$ and $i$) **then**
    $\mathbf{T_i^0} = x_{\lceil m/2 \rceil + \lfloor i/2 \rfloor}$;
  **else**
    $\mathbf{T_i^0} = \emptyset$;
  **end if**
  $l = i + 1$;
  **for** $k = 1$ to $\lfloor log_2 m \rfloor$ **do**
    **if** $\lfloor (m\text{–}1\text{–}i)/2^k \rfloor \; mod \; 2 = 1$ $\{\exists \; \mathbf{T_i^k}$ at level $k\}$ **then**
      $\mathbf{T_i^k} = \sum_{h=l}^{\pi} z_{h,m-h+i}$; $l = \pi + 1$;
    **else**
      $\mathbf{T_i^k} = \emptyset$;
    **end if**
  **end for**
**end for**

A characteristic of the previous representation is that the coefficients $(s_\rho^i, \ldots, s_1^i, s_0^i)$ and $(t_\rho^i, \ldots, t_1^i, t_0^i)$ are given by the binary representations of the subindex $i$ for $\mathbf{S_i}$ and of the value $m-1-i$ for $\mathbf{T_i}$, respectively. This can be deduced from the expressions of $\mathbf{S_i}$ and $\mathbf{T_i}$ defined in (1) and (2). For example, from (1) it can be observed that $\mathbf{S_i}$ is given by the sum of $i$ product terms $a_k b_l$. As any number $i$ can be given as a sum of powers of 2, then $\mathbf{S_i}$ can also be given as a sum of powers of 2 of product terms $a_k b_l$. The addition of $2^j$ products was previously denoted as $\mathbf{S_i^j}$. Therefore, in the notation $\mathbf{S_i} = s_\rho^i \mathbf{S_i^\rho} + \ldots + s_0^i \mathbf{S_i^0}$, the coefficients $(s_\rho^i, \ldots, s_1^i, s_0^i)$ correspond with the binary representation of $i$. A similar reasoning can be done for $\mathbf{T_i}$ considering that $\mathbf{T_i}$ is given by the sum of $m-1-i$ product terms $a_k b_l$. Furthermore, in order to reduce the number of XORs needed for the computation of the product, common terms appearing in several coefficients can also be shared. These common terms correspond to the addition of consecutive $\mathbf{S_i}$ and $\mathbf{T_i}$ terms, i.e., $(\mathbf{S_i} + \mathbf{S_{i+1}})$ and $(\mathbf{T_i} + \mathbf{T_{i+1}})$, that lead to the addition of terms $(\mathbf{S_i^l} + \mathbf{S_i^l})$ and $(\mathbf{T_i^l} + \mathbf{T_i^l})$, respectively, for different levels $l$ determined by the binary representations of the subindex $i$ (for $\mathbf{S_i}$)

and $m-1-i$ (for $\mathbf{T_i}$). The addition of any pair of terms in level $l$ creates a new term in level $l+1$. From Table I, it can be noted that for the coefficients of the multiplier only common additions $(\mathbf{T_i}+\mathbf{T_{i+1}})$ can be found. Using the binary representations of $m-1-i$, it can be observed that for *even* $m$, the common sums are $(\mathbf{T_i}+\mathbf{T_{i+1}})$ for $i=0,\ldots,m-4$, while that for *odd* $m$, the common terms are $(\mathbf{T_i}+\mathbf{T_{i+1}})$ for $i=1,\ldots,m-4$. The occurrence of these common groups in the coefficients of the product is studied in Appendix B.

The algorithm for the computation of the new proposed multiplication is given in Algorithm 3. In the first *for* loop, the common terms to be shared $(\mathbf{T_i^l}+\mathbf{T_{i+1}^l})$ are created, where $consecutive^*$ refers to that for *even* $m$, the subindex $i$ ranges from 0 to $m-4$, while that for *odd* $m$, $i$ ranges from 1 to $m-4$. For each coefficient in Table I, the outer *for* loop processes (for each level $l=0,\ldots,\lfloor log_2 m\rfloor$) the initial $\mathbf{S_i^l}$ and $\mathbf{T_i^l}$ terms, creating new $(l+1)$-level terms and sharing common terms (if any). The *while* loop processes terms from level $\lfloor log_2 m\rfloor+1$ to a level $L$ with only two terms, in such a way that the maximum level will be $L+1$ for the given coefficient. The execution of the algorithm will provide the coefficients of the product. The above new method of multiplication is clarified with the following example.

**Algorithm 3** Computation of the product for $GF(2^m)$.

compute $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ terms (using Algorithms 1 and 2)
**for** $l=0$ to $\lfloor log_2 m\rfloor$ **do**
  find $consecutive^*$ $l$-level terms $\mathbf{T_i^l}$ and $\mathbf{T_{i+1}^l}$;
  create common terms $(\mathbf{T_i^l}+\mathbf{T_{i+1}^l})$ in level $l+1$;
**end for**
**for** each coefficient $c_i$ in Table I **do**
  **for** $l=0$ to $\lfloor log_2 m\rfloor$ **do**
    **if** $\exists$ $consecutive^*$ $l$-level terms $\mathbf{T_i^l}$ and $\mathbf{T_{i+1}^l}$ **then**
      share common $(l+1)$-level terms $(\mathbf{T_i^l}+\mathbf{T_{i+1}^l})$;
    **end if**
    **for** the remaining $l$-level terms **do**
      sum $l$-level terms in pairs to create $(l+1)$-level terms;
      **if** $\exists$ a non-paired $l$-level term **then**
        consider the term as a $(l+1)$-level term
      **end if**
    **end for**
  **end for**
  $l=l+1$;
  **while** the number of $l-$ level terms $\geq 2$ **do**
    sum $l$-level terms in pairs to create $(l+1)$-level terms;
    **if** $\exists$ a non-paired $l$-level term **then**
      consider the term as a $(l+1)$-level term
    **end if**
    $l=l+1$;
  **end while**
**end for**

### A. Multiplication Example over $GF(2^{14})$

Let us consider the product $C$ of two elements $A$ and $B$ in $GF(2^{14})$ generated by the type II irreducible pentanomial $f(y)=y^{14}+y^6+y^5+y^4+1$. The $\mathbf{S_i}$ and $\mathbf{T_i}$ functions can be computed using (1), (2) and are given in Table II. In this table, the $\mathbf{S_i}$ and $\mathbf{T_i}$ functions are the sum of the $x_k$ and $z_{i,j}$ terms in the $i$-th row given in the second column. This column is divided into four subcolumns labeled as $2^0$, $2^1$, $2^2$, and $2^3$ that represent the number of product terms $a_hb_l$ involved in each subcolumn. For example, $\mathbf{S_{13}}=x_6+(z_{0,12}+z_{1,11})+(z_{2,10}+z_{3,9}+z_{4,8}+z_{5,7})$, where $x_6$ involves $1=2^0$ product term $(a_6b_6)$, $(z_{0,12}+z_{1,11})$ involves the addition of $4=2^2$ terms $((a_0b_{12}+a_{12}b_0)+(a_1b_{11}+a_{11}b_1))$ and $(z_{2,10}+z_{3,9}+z_{4,8}+z_{5,7})$ is the sum of $8=2^3$ product terms $((a_2b_{10}+a_{10}b_2)+(a_3b_9+a_9b_3)+(a_4b_8+a_8b_4)+(a_5b_7+a_7b_5))$. The term $\mathbf{S_{13}}$ can then be represented in the form $\mathbf{S_{13}}=1.\mathbf{S_{13}^0}+0.\mathbf{S_{13}^1}+1.\mathbf{S_{13}^2}+1.\mathbf{S_{13}^3}$ where $\mathbf{S_{13}^0}$, $\mathbf{S_{13}^1}$, $\mathbf{S_{13}^2}$ and $\mathbf{S_{13}^3}$ stand for the terms with $2^0$, $2^1$, $2^2$ and $2^3$ product terms, respectively. In this case, $\mathbf{S_{13}}$ has not the term $\mathbf{S_{13}^1}$ because it is associated with the binary coefficient 0. The above representation is given in the third column in Table II, where the binary coefficients $(s_0^i,s_1^i,s_2^i,s_3^i)$ and $(t_0^i,t_1^i,t_2^i,t_3^i)$ associated with the terms $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$, respectively, are given. It can be noted that the binary coefficients ordered in the form $(s_3^i,s_2^i,s_1^i,s_0^i)$ for $\mathbf{S_i}$ correspond with the binary representation of the subindex $i$ while that the binary coefficients $(t_3^i,t_2^i,t_1^i,t_0^i)$ for $\mathbf{T_i}$ correspond with the binary representation of $m-1-i$. The fourth column in Table II includes these binary representations. For example, the term $\mathbf{S_{13}}$ corresponds with the binary vector (1101) while that $\mathbf{T_2}$ corresponds with (1011) that is the binary representation of the value $14-1-2=11$ (in this example with $m=14$). It can be observed that the $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ terms given in Table II can also be computed using Algorithm 1 and Algorithm 2.

In order to reduce the space complexity of the multiplier, common terms appearing in several coefficients can also be shared. It can be observed in Table II that consecutive $\mathbf{S_i}$ and $\mathbf{T_i}$ terms have $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ terms with the same level $j$. For example, $\mathbf{S_{10}}$ and $\mathbf{S_{11}}$ have 1-level terms $\mathbf{S_{10}^1}$ and $\mathbf{S_{11}^1}$ and 3-level terms $\mathbf{S_{10}^3}$ and $\mathbf{S_{11}^3}$, respectively. The addition of $\mathbf{S_{10}}$ and $\mathbf{S_{11}}$ then implies the sums $\mathbf{S_{10}^1}+\mathbf{S_{11}^1}$ and $\mathbf{S_{10}^3}+\mathbf{S_{11}^3}$ that give rise to 2-level and 4-level complete binary trees of XOR gates, respectively. Therefore, the group given by the addition of these two functions $\mathbf{S_{10}}$ and $\mathbf{S_{11}}$ can reduce the complexity. In Table II the groups that can be found are represented by shadowed cells with the same color. The $\mathbf{S}$ groups are $(\mathbf{S_2},\mathbf{S_3})$, $(\mathbf{S_4},\mathbf{S_5})$, $(\mathbf{S_6},\mathbf{S_7})$, $(\mathbf{S_8},\mathbf{S_9})$, $(\mathbf{S_{10}},\mathbf{S_{11}})$ and $(\mathbf{S_{12}},\mathbf{S_{13}})$, while that the $\mathbf{T}$ groups are $(\mathbf{T_0},\mathbf{T_1})$, $(\mathbf{T_2},\mathbf{T_3})$, $(\mathbf{T_4},\mathbf{T_5})$, $(\mathbf{T_6},\mathbf{T_7})$, $(\mathbf{T_8},\mathbf{T_9})$ and $(\mathbf{T_{10}},\mathbf{T_{11}})$. The sum of the $\mathbf{S_i}$ and $\mathbf{T_i}$ terms that appear in the coefficients of the product must be done using the above groups in order to optimize the implementation.

The coefficients of the product are given in Table III using Table I for this $GF(2^{14})$ irreducible pentanomial. The previous

TABLE II
$\mathbf{S_i}$ AND $\mathbf{T_i}$ FUNCTIONS FOR $GF(2^{14})$

| | $2^0$ | $2^1$ | $2^2$ | $2^3$ | $\mathbf{S_i^0}$ | $\mathbf{S_i^1}$ | $\mathbf{S_i^2}$ | $\mathbf{S_i^3}$ | binary |
|---|---|---|---|---|---|---|---|---|---|
| $\mathbf{S_1}$ | $x_0$ | | | | 1 | 0 | 0 | 0 | 0001 |
| $\mathbf{S_2}$ | | $z_{0,1}$ | | | 0 | 1 | 0 | 0 | 0010 |
| $\mathbf{S_3}$ | $x_1$ | $z_{0,2}$ | | | 1 | 1 | 0 | 0 | 0011 |
| $\mathbf{S_4}$ | | | $(z_{0,3}+z_{1,2})$ | | 0 | 0 | 1 | 0 | 0100 |
| $\mathbf{S_5}$ | $x_2$ | | $(z_{0,4}+z_{1,3})$ | | 1 | 0 | 1 | 0 | 0101 |
| $\mathbf{S_6}$ | | $z_{0,5}$ | $(z_{1,4}+z_{2,3})$ | | 0 | 1 | 1 | 0 | 0110 |
| $\mathbf{S_7}$ | $x_3$ | $z_{0,6}$ | $(z_{1,5}+z_{2,4})$ | | 1 | 1 | 1 | 0 | 0111 |
| $\mathbf{S_8}$ | | | | $(z_{0,7}+z_{1,6}+z_{2,5}+z_{3,4})$ | 0 | 0 | 0 | 1 | 1000 |
| $\mathbf{S_9}$ | $x_4$ | | | $(z_{0,8}+z_{1,7}+z_{2,6}+z_{3,5})$ | 1 | 0 | 0 | 1 | 1001 |
| $\mathbf{S_{10}}$ | | $z_{0,9}$ | | $(z_{1,8}+z_{2,7}+z_{3,6}+z_{4,5})$ | 0 | 1 | 0 | 1 | 1010 |
| $\mathbf{S_{11}}$ | $x_5$ | $z_{0,10}$ | | $(z_{1,9}+z_{2,8}+z_{3,7}+z_{4,6})$ | 1 | 1 | 0 | 1 | 1011 |
| $\mathbf{S_{12}}$ | | | $(z_{0,11}+z_{1,10})$ | $(z_{2,9}+z_{3,8}+z_{4,7}+z_{5,6})$ | 0 | 0 | 1 | 1 | 1100 |
| $\mathbf{S_{13}}$ | $x_6$ | | $(z_{0,12}+z_{1,11})$ | $(z_{2,10}+z_{3,9}+z_{4,8}+z_{5,7})$ | 1 | 0 | 1 | 1 | 1101 |
| $\mathbf{S_{14}}$ | | $z_{0,13}$ | $(z_{1,12}+z_{2,11})$ | $(z_{3,10}+z_{4,9}+z_{5,8}+z_{6,7})$ | 0 | 1 | 1 | 1 | 1110 |
| | | | | | $\mathbf{T_i^0}$ | $\mathbf{T_i^1}$ | $\mathbf{T_i^2}$ | $\mathbf{T_i^3}$ | |
| $\mathbf{T_0}$ | $x_7$ | | $(z_{1,13}+z_{2,12})$ | $(z_{3,11}+z_{4,10}+z_{5,9}+z_{6,8})$ | 1 | 0 | 1 | 1 | 1101 |
| $\mathbf{T_1}$ | | | $(z_{2,13}+z_{3,12})$ | $(z_{4,11}+z_{5,10}+z_{6,9}+z_{7,8})$ | 0 | 0 | 1 | 1 | 1100 |
| $\mathbf{T_2}$ | $x_8$ | $z_{3,13}$ | | $(z_{4,12}+z_{5,11}+z_{6,10}+z_{7,9})$ | 1 | 1 | 0 | 1 | 1011 |
| $\mathbf{T_3}$ | | $z_{4,13}$ | | $(z_{5,12}+z_{6,11}+z_{7,10}+z_{8,9})$ | 0 | 1 | 0 | 1 | 1010 |
| $\mathbf{T_4}$ | $x_9$ | | | $(z_{5,13}+z_{6,12}+z_{7,11}+z_{8,10})$ | 1 | 0 | 0 | 1 | 1001 |
| $\mathbf{T_5}$ | | | | $(z_{6,13}+z_{7,12}+z_{8,11}+z_{9,10})$ | 0 | 0 | 0 | 1 | 1000 |
| $\mathbf{T_6}$ | $x_{10}$ | $z_{7,13}$ | $(z_{8,12}+z_{9,11})$ | | 1 | 1 | 1 | 0 | 0111 |
| $\mathbf{T_7}$ | | $z_{8,13}$ | $(z_{9,12}+z_{10,11})$ | | 0 | 1 | 1 | 0 | 0110 |
| $\mathbf{T_8}$ | $x_{11}$ | | $(z_{9,13}+z_{10,12})$ | | 1 | 0 | 1 | 0 | 0101 |
| $\mathbf{T_9}$ | | | $(z_{10,13}+z_{11,12})$ | | 0 | 0 | 1 | 0 | 0100 |
| $\mathbf{T_{10}}$ | $x_{12}$ | $z_{11,13}$ | | | 1 | 1 | 0 | 0 | 0011 |
| $\mathbf{T_{11}}$ | | $z_{12,13}$ | | | 0 | 1 | 0 | 0 | 0010 |
| $\mathbf{T_{12}}$ | $x_{13}$ | | | | 1 | 0 | 0 | 0 | 0001 |

$\mathbf{S}$ and $\mathbf{T}$ groups found in the product coefficients are shadowed in Table III. It can be observed that only one $\mathbf{S_i}$ term appears in each coefficient, so only the $\mathbf{T}$ groups are used. The group $(\mathbf{T_8}, \mathbf{T_9})$ appears in three coefficients ($c_0$, $c_4$ and $c_{13}$) and the groups $(\mathbf{T_0}, \mathbf{T_1})$, $(\mathbf{T_2}, \mathbf{T_3})$, $(\mathbf{T_4}, \mathbf{T_5})$, $(\mathbf{T_6}, \mathbf{T_7})$ and $(\mathbf{T_{10}}, \mathbf{T_{11}})$ appear in two coefficients. This means that only one of each of the above groups must be implemented and therefore the other occurrences of the groups must not be implemented. The number of XOR gates that can be reduced is given by the number of $\mathbf{T_i^j}$ terms in each group. Using Table II it can be observed that the group $(\mathbf{T_0}, \mathbf{T_1})$ involves the sum of two terms $\mathbf{T_0^2} + \mathbf{T_1^2}$ and $\mathbf{T_0^3} + \mathbf{T_1^3}$ and therefore it requires 2 XOR gates. In the same way, the groups $(\mathbf{T_2}, \mathbf{T_3})$, $(\mathbf{T_4}, \mathbf{T_5})$, $(\mathbf{T_6}, \mathbf{T_7})$, $(\mathbf{T_8}, \mathbf{T_9})$ and $(\mathbf{T_{10}}, \mathbf{T_{11}})$ require 2, 1, 2, 1, and 1 XOR gates, respectively. Furthermore, as $(\mathbf{T_8}, \mathbf{T_9})$ appears in three coefficients, then the number of XOR gates that can be reduced will be 2 times the number of XOR gates required, i.e., $2 \cdot 1 = 2$. Therefore, the number of XOR gates that will be reduced for the above groups will be, respectively, $2+2+1+2+2+1 = 10$ XOR. In Section III-B, general expressions will be given in order to compute the number of XOR gates that can be reduced due to the groups.

Using the $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ terms given in Table II for $\mathbf{S_i}$ and $\mathbf{T_i}$, respectively, the coefficients of the product are shown in Table IV. In this table, the sum of terms is accomplished using the rule previously given, i.e., the sum of the functions $\mathbf{S_i}$ and $\mathbf{T_i}$ is performed by grouping the sums of terms with the same $j$-level $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$, starting with the lower levels. In this way, the 0-level initial terms $\mathbf{S_i^0}$ and $\mathbf{T_i^0}$ should be first added in pairs to give rise to new 1-level binary trees (1-level terms), that in turn should be added in pairs with other 1-level terms to give rise to new 2-level complete binary trees and so on. If there is only one $j$-level term (or there is an unpaired $j$-level term), then it should be added with an immediately above $(j+1)$-level term in order to have a new $(j+2)$-level tree. If no such a $(j+1)$-level term exists, then it should be added with a $(j+2)$-level term, and so on. The order of the additions of the terms in Table IV is represented by means of parenthesis. To reduce the number of XORs needed for the computation of the product, common terms appearing in several coefficients can also be shared. This is represented with shadowed boxes that correspond with the previously stated groups.

TABLE III
COEFFICIENTS $c_i$ OF THE PRODUCT FOR $GF(2^{14})$

| | | | |
|---|---|---|---|
| $c_0$ | = | $S_1 + T_0 + T_8 + T_9 + T_{10}$; | |
| $c_1$ | = | $S_2 + T_1 + T_9 + T_{10} + T_{11}$; | A |
| $c_2$ | = | $S_3 + T_2 + T_{10} + T_{11} + T_{12}$; | |
| $c_3$ | = | $S_4 + T_3 + T_{11} + T_{12}$; | |
| $c_4$ | = | $S_5 + T_4 + T_{12} + T_0 + T_8 + T_9 + T_{10}$; | B |
| $c_5$ | = | $S_6 + T_5 + T_0 + T_8 + T_1 + T_{11}$; | |
| $c_6$ | = | $S_7 + T_6 + T_0 + T_8 + T_{10} + T_1 + T_2 + T_{12}$; | C |
| $c_7$ | = | $S_8 + T_7 + T_1 + T_9 + T_{11} + T_2 + T_3$; | D |
| $c_8$ | = | $S_9 + T_8 + T_2 + T_{10} + T_{12} + T_3 + T_4$; | |
| $c_9$ | = | $S_{10} + T_9 + T_3 + T_{11} + T_4 + T_5$; | E |
| $c_{10}$ | = | $S_{11} + T_{10} + T_4 + T_{12} + T_5 + T_6$; | |
| $c_{11}$ | = | $S_{12} + T_{11} + T_5 + T_6 + T_7$; | F |
| $c_{12}$ | = | $S_{13} + T_{12} + T_6 + T_7 + T_8$; | |
| $c_{13}$ | = | $S_{14} + T_7 + T_8 + T_9$; | G |

In order to illustrate the method, the implementation of the coefficient $c_6$ is given in Fig. 1. This coefficient requires the addition of 8 terms, so it is the most complex coefficient and determine the maximum delay of the multiplier (in fact, the $c_{n+2}$ coefficient of a $GF(2^m)$ multiplier given by Type II pentanomials is the most complex one, so it is used to determine the maximum delay complexity). In this figure, the $\mathbf{S_i}$ and $\mathbf{T_i}$ terms are represented by filled circles. These circles correspond to the initial $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ terms given in Table II in such a way that

TABLE IV
COEFFICIENTS $c_i$ OF THE PRODUCT FOR $GF(2^{14})$ WITH $\mathbf{S_i^j}$ AND $\mathbf{T_i^j}$ TERMS

| | | |
|---|---|---|
| $c_0$ | = | $(((\mathbf{S_1^0}+\mathbf{T_0^0})+\mathbf{T_0^2})+\mathbf{T_0^3})+(((\mathbf{T_8^0}+\mathbf{T_{10}^0})+\mathbf{T_{10}^1})+(\mathbf{T_8^2}+\mathbf{T_9^2}))$; |
| $c_1$ | = | $((\mathbf{S_2^1}+\mathbf{T_{10}^0})+(\mathbf{T_{10}^1}+\mathbf{T_{11}^1}))+((\mathbf{T_1^2}+\mathbf{T_9^2})+\mathbf{T_1^3})$; |
| $c_2$ | = | $(((\mathbf{S_3^0}+\mathbf{T_2^0})+\mathbf{S_3^1})+((\mathbf{T_{10}^0}+\mathbf{T_{12}^0})+\mathbf{T_2^1}))+((\mathbf{T_{10}^1}+\mathbf{T_{11}^1})+\mathbf{T_2^3})$; |
| $c_3$ | = | $((\mathbf{S_4^2}+\mathbf{T_3^1})+\mathbf{T_3^3})+(\mathbf{T_{11}^1}+\mathbf{T_{12}^0})$; |
| $c_4$ | = | $(((((\mathbf{S_5^0}+\mathbf{T_0^0})+(\mathbf{T_4^0}+\mathbf{T_8^0}))+((\mathbf{T_{10}^0}+\mathbf{T_{12}^0})+\mathbf{T_{10}^1}))+\mathbf{T_4^3})+((\mathbf{S_5^2}+\mathbf{T_0^2})+\mathbf{T_0^3}))+(\mathbf{T_8^2}+\mathbf{T_9^2})$; |
| $c_5$ | = | $(((((\mathbf{T_0^0}+\mathbf{T_8^0})+\mathbf{S_6^1})+\mathbf{S_6^2})+(\mathbf{T_0^2}+\mathbf{T_1^2}))+(\mathbf{T_0^3}+\mathbf{T_1^3}))+((\mathbf{T_{11}^1}+\mathbf{T_8^2})+\mathbf{T_5^3})$; |
| $c_6$ | = | $(((((\mathbf{S_7^0}+\mathbf{T_0^0})+\mathbf{S_7^1})+\mathbf{S_7^2})+(\mathbf{T_0^2}+\mathbf{T_1^2}))+(\mathbf{T_0^3}+\mathbf{T_1^3}))+$ $(((((\mathbf{T_2^0}+\mathbf{T_6^0})+\mathbf{T_2^1})+\mathbf{T_6^2})+\mathbf{T_2^3})+((((\mathbf{T_8^0}+\mathbf{T_{10}^0})+\mathbf{T_6^1})+\mathbf{T_8^2})+(\mathbf{T_{10}^1}+\mathbf{T_{12}^0})))$; |
| $c_7$ | = | $((((\mathbf{T_2^0}+\mathbf{T_7^1})+\mathbf{T_7^2})+(\mathbf{T_2^1}+\mathbf{T_3^1})+\mathbf{T_1^2}))+(\mathbf{S_8^3}+\mathbf{T_1^3}))+((\mathbf{T_{11}^1}+\mathbf{T_9^2})+(\mathbf{T_2^3}+\mathbf{T_3^3}))$; |
| $c_8$ | = | $(((((\mathbf{S_9^0}+\mathbf{T_2^0})+(\mathbf{T_4^0}+\mathbf{T_8^0}))+(\mathbf{T_2^1}+\mathbf{T_3^1}))+\mathbf{S_9^3})+(\mathbf{T_2^3}+\mathbf{T_3^3}))+((((\mathbf{T_{10}^0}+\mathbf{T_{12}^0})+\mathbf{T_{10}^1})+\mathbf{T_8^2})+\mathbf{T_4^3})$; |
| $c_9$ | = | $((((\mathbf{T_4^0}+\mathbf{T_{12}^0})+\mathbf{T_9^2})+\mathbf{T_3^3})+((\mathbf{S_{10}^1}+\mathbf{T_3^1})+\mathbf{S_{10}^3})+(\mathbf{T_4^3}+\mathbf{T_5^3})$; |
| $c_{10}$ | = | $(((((\mathbf{S_{11}^0}+\mathbf{T_4^0})+\mathbf{S_{11}^1})+\mathbf{T_6^2})+\mathbf{S_{11}^3})+(\mathbf{T_4^3}+\mathbf{T_5^3}))+(((\mathbf{T_6^0}+\mathbf{T_{10}^0})+\mathbf{T_6^1})+(\mathbf{T_{12}^0}+\mathbf{T_{10}^1}))$; |
| $c_{11}$ | = | $(((\mathbf{T_6^1}+\mathbf{T_7^1})+\mathbf{S_{12}^2})+(\mathbf{T_6^2}+\mathbf{T_7^2}))+(\mathbf{S_{12}^3}+\mathbf{T_5^3}))+(\mathbf{T_6^0}+\mathbf{T_{11}^1})$; |
| $c_{12}$ | = | $((((\mathbf{S_{13}^0}+\mathbf{T_6^0})+(\mathbf{T_8^0}+\mathbf{T_{12}^0}))+\mathbf{S_{13}^2})+\mathbf{S_{13}^3})+((\mathbf{T_6^1}+\mathbf{T_7^1})+\mathbf{T_8^2})+(\mathbf{T_6^2}+\mathbf{T_7^2}))$; |
| $c_{13}$ | = | $((\mathbf{S_{14}^1}+\mathbf{S_{14}^2})+\mathbf{S_{14}^3})+(((\mathbf{T_7^1}+\mathbf{T_8^0})+\mathbf{T_7^2})+(\mathbf{T_8^2}+\mathbf{T_9^2}))$; |

$\mathbf{S_7^0}+\mathbf{S_7^1}+\mathbf{S_7^2}$, $\mathbf{T_0}=\mathbf{T_0^0}+\mathbf{T_0^2}+\mathbf{T_0^3}$, $\mathbf{T_1}=\mathbf{T_1^2}+\mathbf{T_1^3}$, $\mathbf{T_2}=\mathbf{T_2^0}+\mathbf{T_2^1}+\mathbf{T_2^3}$, $\mathbf{T_6}=\mathbf{T_6^0}+\mathbf{T_6^1}+\mathbf{T_6^2}$, $\mathbf{T_8}=\mathbf{T_8^0}+\mathbf{T_8^2}$, $\mathbf{T_{10}}=\mathbf{T_{10}^0}+\mathbf{T_{10}^1}$ and $\mathbf{T_{12}}=\mathbf{T_{12}^0}$. Vertical dashed lines represent the level of XOR binary trees. For example, the term $\mathbf{S_7^2}$ in line 2 represent the 2-level binary tree $\mathbf{S_7^2}=(z_{1,5}+z_{2,4})=(a_1b_5+a_5b_1)+(a_2b_4+a_4b_2)$. Circles enclosed within ellipses represent the terms of the corresponding $\mathbf{S_i}$ and $\mathbf{T_i}$ functions. For example, the $\mathbf{T_0}$ function is given by the three initial terms $\mathbf{T_0^0}$, $\mathbf{T_0^2}$ and $\mathbf{T_0^3}$. Furthermore, the gray color XOR trees represent the group $(\mathbf{T_0},\mathbf{T_1})$ that can be shared in several coefficients (in this case, the group correspond with the additions $\mathbf{T_0^2}+\mathbf{T_1^2}$ and $\mathbf{T_0^3}+\mathbf{T_1^3}$). It can be observed in Fig. 1 that the addition of terms follows the rule previously given, starting with 0-level terms and ascending in the construction of binary XOR trees. For example, the addition of the initial 0-terms $\mathbf{S_7^0}$ and $\mathbf{T_0^0}$ gives rise to a new 1-level term (a new XOR in level 1) that in turn is added with the initial 1-term $\mathbf{S_7^1}$ to give rise to a new 2-level XOR term and so on. In this example, $c_6$ can be constructed with a 6-level binary XOR tree so the delay complexity of the multiplier is given by $T_{\mathrm{A}}+6T_{\mathrm{X}}$, where $T_{\mathrm{A}}$ and $T_{\mathrm{X}}$ represent the delay of 2-input AND and XOR gates, respectively. The $T_{\mathrm{A}}$ delay corresponds to the 0-level $a_ib_j$ products of the coefficients of $A$ and $B$. It must be noted that the best delay complexity for this multiplier given by other similar methods in the literature is $T_{\mathrm{A}}+7T_{\mathrm{X}}$.

The space complexity (number of AND and XOR gates) can also be computed. The number of AND gates is given by all the different products $a_ib_j$, with $i,j\in[0,\ldots,m-1]$. This number can be computed using (1), (2) and for this $GF(2^{14})$ example is given as 196 AND gates (see also Table II). It is proved in Appendix B that the number of AND gates for a $GF(2^m)$ multiplier is $m^2$. The XOR gates can be computed as the sum of XOR gates in the initial $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ terms (as given in Table II) plus the number of new XOR gates generated in the coefficients (as given in Table IV) minus the number of XOR gates due to the groups shared among coefficients. The $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ terms perform the XOR of $2^j$ product terms, therefore the number of XORs is $2^j-1$. In this example, there are 7 $\mathbf{S_i^0}$, $\mathbf{S_i^1}$, $\mathbf{S_i^2}$ and $\mathbf{S_i^3}$ terms each. Therefore the number of XOR gates in the initial $\mathbf{S_i^j}$ terms

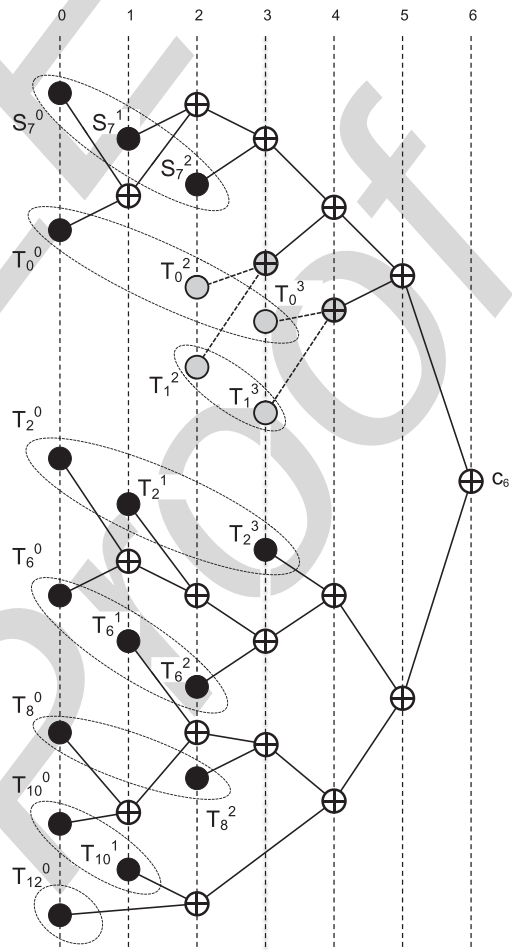


Fig. 1. Implementation of coefficient $c_6$.

will be $7\cdot(2^0-1)+7\cdot(2^1-1)+7\cdot(2^2-1)+7\cdot(2^3-1)=77$ XOR. There are also 7 $\mathbf{T_i^0}$ terms and 6 $\mathbf{T_i^1}$, $\mathbf{T_i^2}$ and $\mathbf{T_i^3}$ terms each, so the number of XOR gates in the initial $\mathbf{T_i^j}$ terms is $7\cdot(2^0-1)+6\cdot(2^1-1)+6\cdot(2^2-1)+6\cdot(2^3-1)=66$ XOR. The number of new XOR gates generated in the coefficients for the sum of $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ terms can be found in Table IV and in this

case is 134 XOR. Finally the number of XORs due to the groups shared among coefficients were previously computed and it was found to be 10 XOR. Therefore the total number of XOR of this multiplier is $77 + 66 + 134 - 10 = 267$ XOR.

### B. Complexity Analysis

General expressions for time and space complexities for the $GF(2^m)$ multiplier are given in this subsection.

*1) Time Complexity:* The coefficients in Table I have been divided into seven sections, depending on the number of $\mathbf{S_i}$ and $\mathbf{T_i}$ terms in the sums. The first section Ⓐ(from $c_0$ to $c_{n-2}$) has 5 terms; section Ⓑwith $c_{n-1}$, $c_n$ and $c_{n+1}$ has 4, 7 and 6 terms, respectively; section Ⓒ($c_{n+2}$ to $c_{2n-2}$) has 8 terms; sections Ⓓ ($c_{2n-1}, c_{2n}$) and Ⓔ($c_{2n+1}, c_{2n+2}$) have 7 and 6 terms, respectively; section Ⓕ($c_{2n+3}$ to $c_{m-2}$) has 5 terms; and finally section Ⓖ($c_{m-1}$) has 4 terms. It can be observed in Table II that the term $\mathbf{T_0}$ has the highest complexity among $\mathbf{T_i}$ terms. The coefficient $c_{n+2}$, that is included in section Ⓒwith the maximum number of terms (8), includes this complex term $\mathbf{T_0}$. Therefore, $c_{n+2}$ is the most complex coefficient of a $GF(2^m)$ multiplier given by Type II irreducible pentanomials and it will be used to determine the highest delay of the multiplier.

In order to do that, the complexity of the $\mathbf{S_i}$ and $\mathbf{T_i}$ terms must be determined. Their complexity depends on the number of the initial $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ terms they have. These terms can be represented as $\mathbf{S_i} = s_\rho^i \mathbf{S_i^\rho} + \ldots + s_0^i \mathbf{S_i^0}$ and $\mathbf{T_i} = t_\rho^i \mathbf{T_i^\rho} + \ldots + t_0^i \mathbf{T_i^0}$ for a given finite field $GF(2^m)$, where $s_j^i, t_j^i \in GF(2)$ and $\rho = \lfloor log_2 m \rfloor$. Therefore, the coefficients $s_j^i, t_j^i$ determine if the corresponding $\mathbf{S_i^j}$, $\mathbf{T_i^j}$ appear in $\mathbf{S_i}$ and $\mathbf{T_i}$, respectively. As previously proved, the coefficients $(s_\rho^i, \ldots, s_1^i, s_0^i)$ and $(t_\rho^i, \ldots, t_1^i, t_0^i)$ are given by the binary representations of the subindex $i$ for $\mathbf{S_i}$ and by the value $m-1-i$ for $\mathbf{T_i}$, respectively. Therefore the study of the number of $\mathbf{T_i^j}$ terms in $\mathbf{T_i}$ can be reduced to the study of $\mathbf{S_i^j}$ terms in $\mathbf{S_i}$ using the equivalence (only in relation to the number of terms) $\mathbf{T_i} \equiv \mathbf{S_{m-1-i}}$. For the most complex coefficient $c_{n+2}$, this equivalence results in that $\mathbf{T_{n+2}} \equiv \mathbf{S_{m-n-3}}$, $\mathbf{T_0} \equiv \mathbf{S_{m-1}}$, $\mathbf{T_{z-2}} \equiv \mathbf{S_{n+1}}$, $\mathbf{T_z} \equiv \mathbf{S_{n-1}}$, $\mathbf{T_1} \equiv \mathbf{S_{m-2}}$, $\mathbf{T_2} \equiv \mathbf{S_{m-3}}$ and $\mathbf{T_{z+2}} \equiv \mathbf{S_{n-3}}$, where $z = m - n$, so $c_{n+2}$ will be equivalent to $c_{n+2} \equiv \mathbf{S_{n+3}} + \mathbf{S_{z-3}} + \mathbf{S_{m-1}} + \mathbf{S_{n+1}} + \mathbf{S_{n-1}} + \mathbf{S_{m-2}} + \mathbf{S_{m-3}} + \mathbf{S_{n-3}}$. The binary representation of $n+3$, $z-3$, $m-1$, $n+1$, $n-1$, $m-2$, $m-3$ and $n-3$ must then be determined. The binary configuration of a number $x$ can be given by the expression

$$x = \sum_{i=0}^{\lfloor log_2 x \rfloor} \left( \left\lfloor \frac{x}{2^i} \right\rfloor \ mod\ 2 \right) \cdot 2^i. \tag{3}$$

The value $\lfloor x/2^i \rfloor \ mod\ 2$ determines if the binary representation of $x$ has a 1 in the position with weight $2^i$ in such a way that if$\lfloor x/2^i \rfloor$ has an *even* value, then $x$ has a 0 while that if $\lfloor x/2^i \rfloor$ has an *odd* value, then $x$ has a 1. A 1 in the position with weight $2^i$ for the binary representation of $x$ will represent that the terms $\mathbf{S_x}$, $\mathbf{T_{m-1-x}}$ have a term $\mathbf{S_x^i}$, $\mathbf{T_{m-1-x}^i}$ that is the sum of $2^i$ product terms and that is implemented by means of a binary XOR tree with depth $i$ (an $i$-level binary tree).

In order to compute the depth of the binary tree of XOR gates in $GF(2^m)$ given by the coefficient $c_{n+2}$, the number of total terms in the $\lfloor log_2 m \rfloor$-level must first be determined. The initial levels for a given $m$ are $0, 1, 2, \ldots, \lfloor log_2 m \rfloor$. For a given level $i$, the number of new XOR terms that will result in level $i+1$ due to the addition in pairs of the $i$-level terms is given by $\lceil number\ of\ i\text{-}level\ terms/2 \rceil$. For example, in Fig. 1 there are seven 0-level terms ($\mathbf{S_7^0}, \mathbf{T_0^0}, \mathbf{T_2^0}, \mathbf{T_6^0}, \mathbf{T_8^0}, \mathbf{T_{10}^0}, \mathbf{T_{12}^0}$) whose sum gives rise to the four 1-level XOR terms ($\mathbf{S_7^0} + \mathbf{T_0^0}$), ($\mathbf{T_2^0} + \mathbf{T_6^0}$), ($\mathbf{T_8^0} + \mathbf{T_{10}^0}$) besides the term $\mathbf{T_{12}^0}$, that can also be considered as a 1-level term (in order to be added to $\mathbf{T_{10}^1}$ and result in a new 2-level XOR term).

Let $\mu_j$ be the number of initial terms $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ in level $j$. This number will be given by the terms $\mathbf{S_i}$ in the previously computed equivalent expression (in relation to the number of terms) $c_{n+2} \equiv \mathbf{S_{n+3}} + \mathbf{S_{z-3}} + \mathbf{S_{m-1}} + \mathbf{S_{n+1}} + \mathbf{S_{n-1}} + \mathbf{S_{m-2}} + \mathbf{S_{m-3}} + \mathbf{S_{n-3}}$. In order to do that, the binary representation of $n+3$, $z-3$, $m-1$, $n+1$, $n-1$, $m-2$, $m-3$, and $n-3$ must then be determined. Using (3), the value $\lfloor x/2^j \rfloor \ mod\ 2$ determines if the term $\mathbf{S_x}$ has an initial term $\mathbf{S_x^j}$ in the position with weight $2^j$, i.e., in level $j$. Representing $\lfloor x/2^j \rfloor \ mod\ 2$ as $\lfloor x/2^j \rfloor^*$, then the number of initial terms $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ in level $j$ (i.e., $\mu_j$) can be computed as follows:

$$\left\lfloor \frac{m-1}{2^j} \right\rfloor^* + \left\lfloor \frac{m-2}{2^j} \right\rfloor^* + \left\lfloor \frac{m-3}{2^j} \right\rfloor^* + \left\lfloor \frac{n+3}{2^j} \right\rfloor^* + \left\lfloor \frac{n+1}{2^j} \right\rfloor^* + \left\lfloor \frac{n-3}{2^j} \right\rfloor^* + \left\lfloor \frac{n-1}{2^j} \right\rfloor^* + \left\lfloor \frac{z-3}{2^j} \right\rfloor^* \tag{4}$$

It must be noted that in (4) the fourth addend $\lfloor (n+3)/2^j \rfloor^*$ corresponds with the *real* $\mathbf{S_{n+3}}$ term, while that the rest of addends correspond with the equivalence previously given $\mathbf{S_{m-1-i}} \equiv \mathbf{T_i}$. Using (4), the number of initial terms $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ for the coefficient $c_6$ given in the example in Section III-A can be computed. The number of initial terms in level 2, for example, will be $\mu_2 = \lfloor 13/4 \rfloor^* + \lfloor 12/4 \rfloor^* + \lfloor 11/4 \rfloor^* + \lfloor 7/4 \rfloor^* + \lfloor 5/4 \rfloor^* + \lfloor 1/4 \rfloor^* + \lfloor 3/4 \rfloor^* + \lfloor 7/4 \rfloor^* = 1+1+0+1+1+0+0+1 = 5$ corresponding with the initial terms $\mathbf{S_{13}^2} \equiv \mathbf{T_0^2}$, $\mathbf{S_{12}^2} \equiv \mathbf{T_1^2}$, $\mathbf{S_7^2}$, $\mathbf{S_5^2} \equiv \mathbf{T_8^2}$, $\mathbf{S_7^2} \equiv \mathbf{T_6^2}$, respectively, that are represented in Fig. 1 as filled (black and gray) circles.

If $\mu_0, \mu_1, \ldots, \mu_{\lfloor log_2 m \rfloor}$ denote the number of initial $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ terms in levels $0, 1, \ldots, \lfloor log_2 m \rfloor$, respectively, then the total number of terms in the $\lfloor log_2 m \rfloor$-level (denoted by $M_{\lfloor log_2 m \rfloor}$) will be the addition of the initial terms in that level $\mu_{\lfloor log_2 m \rfloor}$ plus the terms created due to the addition of terms in lower levels. In Section A of Appendix A, it is proved that these terms created in level $\lfloor log_2 m \rfloor$ due to the addition of terms in levels $0, 1, \ldots, \lfloor log_2 m \rfloor - 1$ is given by the expression:

$$\left\lceil \frac{\mu_0 + 2\mu_1 + 2^2\mu_2 + \ldots + 2^{\lfloor log_2 m \rfloor - 1}\mu_{\lfloor log_2 m \rfloor - 1}}{2^{\lfloor log_2 m \rfloor}} \right\rceil \tag{5}$$

Therefore, the total number $M_{\lfloor log_2 m \rfloor}$ of terms in the $\lfloor log_2 m \rfloor$-level will be the sum of $\mu_{\lfloor log_2 m \rfloor}$ plus the expression in (5). In Appendix A it is proved that this addition is:

$$M_{\lfloor log_2 m \rfloor} = \left\lceil \frac{4m + 3n - 9}{2^{\lfloor log_2 m \rfloor}} \right\rceil \tag{6}$$

The sum in pairs of the terms $M_{\lfloor log_2 m \rfloor}$ determined in (6) will determine the final level reached to compute the $c_{n+2}$ coefficient. Therefore the number of XOR levels needed to compute this coefficient will be $\lfloor log_2 m \rfloor + \lceil log_2 M_{\lfloor log_2 m \rfloor} \rceil$. Finally, the highest delay of the $GF(2^m)$ multiplier based on type II pentanomials given by the $c_{n+2}$ coefficient is:

$$T_A + \left( \lfloor log_2 m \rfloor + \left\lceil log_2 \left\lceil \frac{4m + 3n - 9}{2^{\lfloor log_2 m \rfloor}} \right\rceil \right\rceil \right) T_X \tag{7}$$

In order to compare this time complexity with other multipliers found in the literature, in Section B of Appendix A the following upper bound is derived for the XOR delay of the multiplier:

$$\lfloor log_2 m \rfloor + \lceil log_2 M_{\lfloor log_2 m \rfloor} \rceil \leq 3 + \lceil log_2(m+1) \rceil \quad (8)$$

*2) Area Complexity:* In order to determine the area complexity of the PB multiplier given in Table I, the number of AND and XOR gates of the $\mathbf{S_i}$ and $\mathbf{T_i}$ terms must be known. In this work, these terms have been considered as a sum of $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ terms, in such a way that $\mathbf{S_i} = s_\rho^i \mathbf{S_i^\rho} + \ldots + s_0^i \mathbf{S_i^0}$ and $\mathbf{T_i} = t_\rho^i \mathbf{T_i^\rho} + \ldots + t_0^i \mathbf{T_i^0}$ for a given finite field $GF(2^m)$, where $s_j^i, t_j^i \in GF(2)$ and $\rho = \lfloor log_2 m \rfloor$. In Table I, the coefficients of the product are given as sums of $\mathbf{S_i}$ and $\mathbf{T_i}$ terms where their corresponding components $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ are considered as individual terms when performing the sum. It can be observed that the $\mathbf{S_i}$ terms, $i = 1, 2, \ldots, m$, appears only once while that the $\mathbf{T_i}$ terms, $i = 0, 1, \ldots, m-2$, appear several times.

One way to determine the number of AND and XOR gates of the $\mathbf{S_i}$ and $\mathbf{T_i}$ terms is to count the number of AND and XOR gates given by the sum of terms $x_k$ and $z_{i,j}$ in (1), (2). In this way, we compute the total number of AND gates of the multiplier, the XORs of the $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ terms, the XORs needed for the sum of all the $\mathbf{S_i^j}$ terms of $\mathbf{S_i}$ and the XORs needed for *one* sum of the $\mathbf{T_i^j}$ terms of $\mathbf{T_i}$, i.e., we count the XORs due to the contribution of *all* the $\mathbf{S_i}$ terms and of *one* occurrence of the $\mathbf{T_i}$ terms. If a term $\mathbf{T_i}$ appears $p_i$ times in the additions given for the coefficients in Table I, then the other $p_i - 1$ occurrences are taken into account by computing the number of XORs needed for the sum of the $\mathbf{T_i^j}$ terms of $\mathbf{T_i}$ and multiplying it by $p_i - 1$. This must be done for each $\mathbf{T_i}$, $i = 0, 1, \ldots, m-2$. To determine the area complexity of the PB multiplier, the number of XOR gates needed for the sum of the $\mathbf{S_i}$ and $\mathbf{T_i}$ terms in the product coefficients of Table I and the number of shared groups $(\mathbf{T_i}, \mathbf{T_j})$ that appear in the product coefficients should also be computed. This number of groups must be subtracted from the previous XOR gates computed. Therefore, the following figures must be computed to obtain the XORs of the multiplier:

① The number of XOR gates given by $\mathbf{S_i}$ and $\mathbf{T_i}$ in (1), (2).

② The number of XOR gates needed for the sum of the $\mathbf{S_i}$ and $\mathbf{T_i}$ terms in the product coefficients.

③ For each $\mathbf{T_i}$, the number $p_i$ of times that $\mathbf{T_i}$ appears in Table I and the number $\Theta_i$ of XORs needed for the sum of the $\mathbf{T_i^j}$ terms for $\mathbf{T_i}$ must be determined. Then the XOR gates given by $\sum_{i=0}^{m-2}(p_i - 1) \cdot \Theta_i$ must be computed.

④ The number of XOR gates given by the shared groups $(\mathbf{T_i}, \mathbf{T_j})$ that appear in the product coefficients.

The XOR gates of the multiplier will be ① + ② + ③ − ④. In Appendix B the following values have been computed:

- The total number of AND gates of the multiplier is $m^2$.
- The number ① of XOR gates given by $\mathbf{S_i}$ and $\mathbf{T_i}$ in (1), (2) is $(m-1)^2$.
- The number ② of XOR gates needed for the sum of the $\mathbf{S_i}$ and $\mathbf{T_i}$ terms in the product coefficients is $4m + 3n - 2$.
- The number ③ of XOR gates can be computed by $\sum_{i=0}^{m-2}(p_i - 1) \cdot \Theta_i = 3 \cdot \Upsilon_{m-1} + 3 \cdot \Upsilon_{n+1} - \Psi_n$, where the number of XOR gates $\Psi_n$ needed for the sum of the $\mathbf{S_n^j}$ terms of $\mathbf{S_n}$ is given by:

$$\Psi_n = \left( n - \sum_{j=1}^{\lfloor log_2 n \rfloor} \left\lfloor \frac{n}{2^j} \right\rfloor \right) - 1 = H_n - 1 \quad (9)$$

where $H_n$ is the *Hamming Weight* of $n$ and where $\Upsilon_h = \sum_{i=1}^{h} \Psi_i$ is given as:

$$\Upsilon_h = \sum_{i=1}^{h} \Psi_i = \frac{h(h-1)}{2} - \sum_{i=1}^{h} \sum_{j=1}^{\lfloor log_2 i \rfloor} \left\lfloor \frac{i}{2^j} \right\rfloor. \quad (10)$$

- The number of XOR gates ④ given by the shared groups $(\mathbf{T_i}, \mathbf{T_j})$ that appear in the product coefficients is:

$$\sum_{i=2,4,6,\ldots}^{\frac{\iota}{\kappa}} H_i + H_{n^\dagger/(n-1)^\ddagger} = \Sigma_m + H_{\dagger/\ddagger} \quad (11)$$

where $\iota$ represents the limit $(m-2)$ of the summatory for *even* $m$, $\kappa$ represents the limit $(m-3)$ for *odd* $m$,$^\dagger$ represents the Hamming Weight of $n$ to be computed for *even* $n$ and $^\ddagger$ the Hamming Weight of $(n-1)$ for *odd* $n$.

Therefore, the XOR gates of the multiplier given by the addition ① + ② + ③ − ④ will be

$$m^2 + 2m + 3n + 3(\Upsilon_{m-1} + \Upsilon_{n+1}) - H_n - \Sigma_m - H_{\dagger/\ddagger} \quad (12)$$

A more compact expression for (12) could not be found. The functions $\Upsilon_m$ and $\Sigma_m = \sum_{i=2,4,6,\ldots} H_i$ could be computed for any value of $m$ using *Maple*. In Table VII the values of these functions for $m \in [8, 100]$ are given. Using Table VII, it can be observed that for the example given in Section III-A with $m = 14$, $n = 4$, the values $\Upsilon_{13} = 12$ and $\Sigma_{14} = 9$. In this example, the values $\Upsilon_5 = 2$ and $H_4 = 1$ can also be computed. Applying the above values to (12) we have $196 + 28 + 12 + 3(12+2) - 1 - 9 - 1 = 267$ XOR gates, matching the result given in Section III-A.

## IV. Comparison With Other PB Multipliers

In Table V the theoretical complexities obtained by the approach here proposed are compared with the best results known to date for bit-parallel polynomial basis multipliers over $GF(2^m)$ generated by type II irreducible pentanomials. In (8) it was proved that $\lfloor log_2 m \rfloor + \lceil log_2 M_{\lfloor log_2 m \rfloor} \rceil \leq 3 + \lceil log_2(m+1) \rceil$. It can also be observed that $3 + \lceil log_2(m+1) \rceil \geq 3 + \lceil log_2(m-2) \rceil$, where $3 + \lceil log_2(m-2) \rceil$ is the best XOR delay found in the literature for this type of bit-parallel multipliers [9]. Simulations have been done using *Maple* that have proved that the delay of our multiplier is less than or equal to the delay in [9], i.e., $\lfloor log_2 m \rfloor + \lceil log_2 M_{\lfloor log_2 m \rfloor} \rceil \leq 3 + \lceil log_2(m-2) \rceil$. From the simulation results, it was found that for the 593 different values of $m$ in the interval $m \in [8, 1000]$ for which an irreducible type II pentanomial exists, the proposed multiplier has the smallest delay in 465 different values of the field size $m$. More specifically, among the type II irreducible pentanomials existent in $m \in [8, 1000]$, there are 477 and 1162 different combinations of $(m, n)$ for which the proposed multiplier has equal and less delay, respectively, than the multiplier in [9]. With respect to area complexity, it was found that the proposed multiplier presents equal number of AND gates in comparison with the other similar multipliers existing in the literature (except for the approach presented in [21]) and a higher number of XOR gates in comparison with the other multipliers. This increased

TABLE V
COMPLEXITIES OF BIT-PARALLEL PB MULTIPLIERS FOR $f(x) = x^m + x^{n+2} + x^{n+1} + x^n + 1$

| | #AND | #XOR | Delay |
|---|---|---|---|
| [13] | $m^2$ | $m^2 + 2m - 3$ | $T_{\rm A} + (6 + \lceil log_2 m \rceil)T_{\rm X}$ |
| [12] | $m^2$ | $m^2 + 2m - 3$ | $T_{\rm A} + (4 + \lceil log_2 (m-1) \rceil)T_{\rm X}$ |
| [18] | $m^2$ | $m^2 + 2m - 3$ | $T_{\rm A} + (4 + \lceil log_2 (m-1) \rceil)T_{\rm X}$ |
| [6] | $m^2$ | $m^2 + m + \lceil \frac{m+n}{2} \rceil - k$ with $k \in [2, 3]$ | $T_{\rm A} + (4 + \lceil log_2 (m-1) \rceil)T_{\rm X}$ |
| [9] | $m^2$ | $m^2 + \frac{3m+6n-\lambda}{2}$ | $T_{\rm A} + (3 + \lceil log_2 (m-2) \rceil)T_{\rm X}$ |
| [21] | $\frac{3m^2+2m-1}{4}$ | $\frac{3m^2+24m+14n+\delta}{4}$ | $T_{\rm A} + (3 + \lceil log_2 (m+1) \rceil)T_{\rm X}$ |
| This work | $m^2$ | $m^2 + 2m + 3n + 3(\Upsilon_{m-1} + \Upsilon_{n+1}) - H_n - \Sigma_m - H_{\dagger/\ddagger}$ | $T_A + \left( \lfloor log_2 m \rfloor + \left\lceil log_2 \left\lceil \frac{4m+3n-9}{2^{\lfloor log_2 m \rfloor}} \right\rceil \right\rceil \right) T_X$ |

$\lambda = 2$ (odd $n$, even $m$), 3 (odd $n$, odd $m$), 4 (even $n$, even $m$) or 5 (even $n$, odd $m$).
$\delta = 35$ (odd $n$, $n \leq (3/8)(m-7)$), 45 (even $n$).
$\dagger = n$ (*even* $n$). $\ddagger = n - 1$ (*odd* $n$).

TABLE VI
COMPLEXITIES OF BIT-PARALLEL PB MULTIPLIERS USING TYPE II PENTANOMIALS FOR THE FIVE RECOMMENDED NIST FIELDS

| | #AND | #XOR | Delay |
|---|---|---|---|
| $f(x) = x^{163} + x^{68} + x^{67} + x^{66} + 1$ | | | |
| [13] | 26569 | 26892 | $T_{\rm A} + 14T_{\rm X}$ |
| [12] | 26569 | 26892 | $T_{\rm A} + 12T_{\rm X}$ |
| [18] | 26569 | 26892 | $T_{\rm A} + 12T_{\rm X}$ |
| [6] | 26569 | 26844 | $T_{\rm A} + 12T_{\rm X}$ |
| [9] | 26569 | 27009 | $T_{\rm A} + 11T_{\rm X}$ |
| [21] | 20008 | 21147 | $T_{\rm A} + 11T_{\rm X}$ |
| This work | 26569 | 28464 | $T_{\rm A} + 10T_{\rm X}$ |
| $f(x) = x^{233} + x^{58} + x^{57} + x^{56} + 1$ | | | |
| [13] | 54289 | 54752 | $T_{\rm A} + 14T_{\rm X}$ |
| [12] | 54289 | 54752 | $T_{\rm A} + 12T_{\rm X}$ |
| [18] | 54289 | 54752 | $T_{\rm A} + 12T_{\rm X}$ |
| [6] | 54289 | 54664 | $T_{\rm A} + 12T_{\rm X}$ |
| [9] | 54289 | 54804 | $T_{\rm A} + 11T_{\rm X}$ |
| [21] | 40833 | 42322 | $T_{\rm A} + 11T_{\rm X}$ |
| This work | 54289 | 56819 | $T_{\rm A} + 11T_{\rm X}$ |
| $f(x) = x^{283} + x^{25} + x^{24} + x^{23} + 1$ | | | |
| [13] | 80089 | 80652 | $T_{\rm A} + 15T_{\rm X}$ |
| [12] | 80089 | 80652 | $T_{\rm A} + 13T_{\rm X}$ |
| [18] | 80089 | 80652 | $T_{\rm A} + 13T_{\rm X}$ |
| [6] | 80089 | 80523 | $T_{\rm A} + 13T_{\rm X}$ |
| [9] | 80089 | 80581 | $T_{\rm A} + 12T_{\rm X}$ |
| [21] | 60208 | 61854 | $T_{\rm A} + 12T_{\rm X}$ |
| This work | 80089 | 82811 | $T_{\rm A} + 11T_{\rm X}$ |
| $f(x) = x^{409} + x^{137} + x^{136} + x^{135} + 1$ | | | |
| [13] | 167281 | 168096 | $T_{\rm A} + 15T_{\rm X}$ |
| [12] | 167281 | 168096 | $T_{\rm A} + 13T_{\rm X}$ |
| [18] | 167281 | 168096 | $T_{\rm A} + 13T_{\rm X}$ |
| [6] | 167281 | 167960 | $T_{\rm A} + 13T_{\rm X}$ |
| [9] | 167281 | 168298 | $T_{\rm A} + 12T_{\rm X}$ |
| [21] | 125665 | 128396 | $T_{\rm A} + 12T_{\rm X}$ |
| This work | 167281 | 172639 | $T_{\rm A} + 11T_{\rm X}$ |
| $f(x) = x^{571} + x^{105} + x^{104} + x^{103} + 1$ | | | |
| [13] | 326041 | 327180 | $T_{\rm A} + 16T_{\rm X}$ |
| [12] | 326041 | 327180 | $T_{\rm A} + 14T_{\rm X}$ |
| [18] | 326041 | 327180 | $T_{\rm A} + 14T_{\rm X}$ |
| [6] | 326041 | 326947 | $T_{\rm A} + 14T_{\rm X}$ |
| [9] | 326041 | 327205 | $T_{\rm A} + 13T_{\rm X}$ |
| [21] | 244816 | 248326 | $T_{\rm A} + 13T_{\rm X}$ |
| This work | 326041 | 332936 | $T_{\rm A} + 12T_{\rm X}$ |

number of XOR gates is due to the *separation* of the *monolithic* functions $\mathbf{S_i}/\mathbf{T_i}$ into the corresponding $\mathbf{S_i^j}/\mathbf{T_i^j}$ terms in order to achieve a reduced delay for multiplication.

In Table VI the complexities of bit-parallel polynomial basis multipliers using type II irreducible pentanomials for the five finite fields $GF(2^m)$ with $m \in \{163, 233, 283, 409, 571\}$ recommended by NIST for ECDSA are presented. From the Table VI, it can be observed that the multiplier here proposed presents the lowest delay except for $GF(2^{233})$, that matches the best delay given in [9].

TABLE VII
COMPUTED VALUES FOR $\Upsilon_m$ and $\Sigma_m$, $m = 8 \ldots 100$

| $m$ | $\Upsilon_m$ | $\Sigma_m$ | $m$ | $\Upsilon_m$ | $\Sigma_m$ | $m$ | $\Upsilon_m$ | $\Sigma_m$ |
|---|---|---|---|---|---|---|---|---|
| 8 | 5 | 4 | 39 | 61 | 37 | 70 | 138 | 85 |
| 9 | 6 | 4 | 40 | 62 | 40 | 71 | 141 | 85 |
| 10 | 7 | 5 | 41 | 64 | 40 | 72 | 142 | 88 |
| 11 | 9 | 5 | 42 | 66 | 42 | 73 | 144 | 88 |
| 12 | 10 | 7 | 43 | 69 | 42 | 74 | 146 | 90 |
| 13 | 12 | 7 | 44 | 71 | 45 | 75 | 149 | 90 |
| 14 | 14 | 9 | 45 | 74 | 45 | 76 | 151 | 93 |
| 15 | 17 | 9 | 46 | 77 | 48 | 77 | 154 | 93 |
| 16 | 17 | 12 | 47 | 81 | 48 | 78 | 157 | 96 |
| 17 | 18 | 12 | 48 | 82 | 52 | 79 | 161 | 96 |
| 18 | 19 | 13 | 49 | 84 | 52 | 80 | 162 | 100 |
| 19 | 21 | 13 | 50 | 86 | 54 | 81 | 164 | 100 |
| 20 | 22 | 15 | 51 | 89 | 54 | 82 | 166 | 102 |
| 21 | 24 | 15 | 52 | 91 | 57 | 83 | 169 | 102 |
| 22 | 26 | 17 | 53 | 94 | 57 | 84 | 171 | 105 |
| 23 | 29 | 17 | 54 | 97 | 60 | 85 | 174 | 105 |
| 24 | 30 | 20 | 55 | 101 | 60 | 86 | 177 | 108 |
| 25 | 32 | 20 | 56 | 103 | 64 | 87 | 181 | 108 |
| 26 | 34 | 22 | 57 | 106 | 64 | 88 | 183 | 112 |
| 27 | 37 | 22 | 58 | 109 | 67 | 89 | 186 | 112 |
| 28 | 39 | 25 | 59 | 113 | 67 | 90 | 189 | 115 |
| 29 | 42 | 25 | 60 | 116 | 71 | 91 | 193 | 115 |
| 30 | 45 | 28 | 61 | 120 | 71 | 92 | 196 | 119 |
| 31 | 49 | 28 | 62 | 124 | 75 | 93 | 200 | 119 |
| 32 | 49 | 32 | 63 | 129 | 75 | 94 | 204 | 123 |
| 33 | 50 | 32 | 64 | 129 | 80 | 95 | 209 | 123 |
| 34 | 51 | 33 | 65 | 130 | 80 | 96 | 210 | 128 |
| 35 | 53 | 33 | 66 | 131 | 81 | 97 | 212 | 128 |
| 36 | 54 | 35 | 67 | 133 | 81 | 98 | 214 | 130 |
| 37 | 56 | 35 | 68 | 134 | 83 | 99 | 217 | 130 |
| 38 | 58 | 37 | 69 | 136 | 83 | 100 | 219 | 133 |

## V. CONCLUSIONS

High-speed algorithms and hardware architectures for computing $GF(2^m)$ multiplication are highly required in several applications, such as coding theory, computer algebra and cryptography. In this paper, a new bit-parallel $GF(2^m)$ polynomial basis multiplier for type II irreducible pentanomials with reduced time-complexity has been presented. The coefficients of the multiplier are computed as a sum of $\mathbf{S_i}$ and $\mathbf{T_i}$ functions given by the addition of product terms of the coefficients of the two operands to be multiplied. In the new approach here proposed, the sum of products in the $\mathbf{S_i}$ and $\mathbf{T_i}$ functions are separated into sums of $2^j$ product terms (corresponding to the initial $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ terms) that can be implemented as binary trees of XOR gates with depth $j$. The sum in pairs of binary trees with the same depth, starting with the lower levels, leads to a reduction of the time complexity of the multiplier. In this paper, a complete multiplication example has been presented. The theoretical complexity analysis has shown that the proposed bit-parallel multiplier presents the lowest delay among the best

results known to date for similar polynomial basis multipliers based on irreducible pentanomials. Simulations have been done that have proved that for the 593 different values of $m$ in the interval $m \in [8, 1000]$ for which an irreducible type II pentanomial exists, the proposed multiplier has the smallest delay in 465 different values of the field size $m$. Furthermore, for the five binary fields recommended by NIST for ECDSA, i.e., $m \in \{163, 233, 283, 409, 571\}$, the multiplier here proposed presents the lowest delay except for $GF(2^{233})$, that matches the best delay given in the literature.

## Appendix A
## Time Complexity

### A. Total Number of Terms in $\lfloor log_2 m \rfloor$-level

Let $\mu_0, \mu_1, \ldots, \mu_{\lfloor log_2 m \rfloor}$ denote the number of initial $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ terms in levels $0, 1, \ldots, \lfloor log_2 m \rfloor$, respectively. As previously stated, for a given level $i$, the number of new XOR terms that will result in level $i+1$ due to the addition in pairs of the $i$-level terms is given by $\lceil number\ of\ i-level\ terms/2 \rceil$. Starting in level 0, then the new terms created in level 1 due to the sum in pairs of the initial terms in level 0, $\mu_0$, will be $\lceil \mu_0/2 \rceil$. The total number of terms in level 1, denoted by $M_1$, will now be $M_1 = \mu_1 + \lceil \mu_0/2 \rceil$. Using the property of *modulo* operation $\lceil x \rceil + n = \lceil x + n \rceil$ for $n$ integer, then we have that $M_1 = \mu_1 + \lceil \mu_0/2 \rceil = \lceil (\mu_0/2) + \mu_1 \rceil = \lceil (\mu_0 + 2\mu_1)/2 \rceil$. Next the new terms created in level 2 due to the sum in pairs of the terms in level 1, $M_1$, will be $\lceil M_1/2 \rceil$. Using the property of *modulo* operation $\lceil \lceil x/m \rceil / n \rceil = \lceil x/(m \cdot n) \rceil$ for positive integers $m, n$ and arbitrary real number $x$, then the new XOR terms created in level 2 will be $\lceil M_1/2 \rceil = \lceil (\lceil (\mu_0 + 2\mu_1)/2 \rceil)/2 \rceil = \lceil (\mu_0 + 2\mu_1)/2^2 \rceil$. The total number of terms in level 2, denoted by $M_2$, will now be $M_2 = \mu_2 + \lceil (\mu_0 + 2\mu_1)/2^2 \rceil = \lceil ((\mu_0 + 2\mu_1)/2^2) + \mu_2 \rceil = \lceil (\mu_0 + 2\mu_1 + 2^2\mu_2)/2^2 \rceil$. Proceeding in the same way we will have that the new XOR terms created in level $\lfloor log_2 m \rfloor$ due to the sum in pairs of the terms in level $\lfloor log_2 m \rfloor - 1$, $M_{\lfloor log_2 m \rfloor - 1}$, will be $\lceil M_{\lfloor log_2 m \rfloor - 1}/2 \rceil = \lceil (\mu_0 + 2\mu_1 + 2^2\mu_2 + \ldots + 2^{\lfloor log_2 m \rfloor - 1}\mu_{\lfloor log_2 m \rfloor - 1})/2^{\lfloor log_2 m \rfloor} \rceil$, that is (5). Finally, the total number $M_{\lfloor log_2 m \rfloor}$ of terms in the $\lfloor log_2 m \rfloor$-level will be the sum of $\mu_{\lfloor log_2 m \rfloor}$ plus the expression in (5), that is:

$$\left\lceil \frac{\mu_0 + 2\mu_1 + 2^2\mu_2 + \ldots + 2^{\lfloor log_2 m \rfloor}\mu_{\lfloor log_2 m \rfloor}}{2^{\lfloor log_2 m \rfloor}} \right\rceil. \quad (13)$$

Now (4) can be used to simplify (13). In (4), the number of initial $\mathbf{S_i^j}$ and $\mathbf{T_i^j}$ terms in level $j$ is given, where $\lfloor x/2^j \rfloor^*$ represents $\lfloor x/2^j \rfloor\ mod\ 2$. The $mod$ operator is defined by the expression $x\ mod\ y = x - y\lfloor x/y \rfloor$, for real $x, y$ $(y \neq 0)$. Therefore $\lfloor x/2^j \rfloor^* = \lfloor x/2^j \rfloor\ mod\ 2 = \lfloor x/2^j \rfloor - 2 \cdot \lfloor \lfloor x/2^j \rfloor / 2 \rfloor = \lfloor x/2^j \rfloor - 2 \cdot \lfloor x/2^{j+1} \rfloor$. Then $\mu_j$ in (4) can be rewritten as $\mu_j = \lfloor (m-1)/2^j \rfloor + \lfloor (m-2)/2^j \rfloor + \lfloor (m-3)/2^j \rfloor + \lfloor (n+3)/2^j \rfloor + \lfloor (n+1)/2^j \rfloor + \lfloor (n-3)/2^j \rfloor + \lfloor (n-1)/2^j \rfloor + \lfloor (z-3)/2^j \rfloor - 2 \cdot (\lfloor (m-1)/2^{j+1} \rfloor + \lfloor (m-2)/2^{j+1} \rfloor + \lfloor (m-3)/2^{j+1} \rfloor + \lfloor (n+3)/2^{j+1} \rfloor + \lfloor (n+1)/2^{j+1} \rfloor + \lfloor (n-3)/2^{j+1} \rfloor + \lfloor (n-1)/2^{j+1} \rfloor + \lfloor (z-3)/2^{j+1} \rfloor) = \Gamma_{2^j} - 2 \cdot \Gamma_{2^{j+1}}$, where we have denoted the sum of the first eight terms as $\Gamma_{2^j}$ and the sum of the eight terms into the parenthesis as $\Gamma_{2^{j+1}}$, so $\mu_j = \Gamma_{2^j} - 2\Gamma_{2^{j+1}}$. It can be observed that $\mu_0 = \Gamma_{2^0} - 2\Gamma_{2^1} = (m-1) + (m-2) + (m-3) + (n+3) + (n+1) + (n-3) + (n-1) + (m-n-3) - 2\Gamma_2 = (4m + 3n - 9) - 2\Gamma_2$. In a similar way, we can then find that

$$\mu_0 = (4m + 3n - 9) - 2\Gamma_2 \quad (14)$$
$$\mu_1 = \Gamma_2 - 2\Gamma_{2^2} \quad (15)$$
$$\ldots \qquad \ldots$$
$$\mu_{\lfloor log_2 m \rfloor} = \Gamma_{2^{\lfloor log_2 m \rfloor}} - 2\Gamma_{2^{\lfloor log_2 m \rfloor + 1}} \quad (16)$$

Using (14)–(16), the numerator of (13) can be simplified as follows:

$$\begin{aligned} &\mu_0 + 2\mu_1 + 2^2\mu_2 + \ldots + 2^{\lfloor log_2 m \rfloor}\mu_{\lfloor log_2 m \rfloor} = \\ &(4m + 3n - 9) - \cancel{2\Gamma_2} + \cancel{2\Gamma_2} - \cancel{2^2\Gamma_{2^2}} + \cancel{2^2\Gamma_{2^2}} - \\ &\cancel{2^3\Gamma_{2^3}} + \cancel{2^3\Gamma_{2^3}} - \ldots - \cancel{2^{\lfloor log_2 m \rfloor}\Gamma_{2^{\lfloor log_2 m \rfloor}}} + \\ &\cancel{2^{\lfloor log_2 m \rfloor}\Gamma_{2^{\lfloor log_2 m \rfloor}}} - 2^{\lfloor log_2 m \rfloor + 1}\Gamma_{2^{\lfloor log_2 m \rfloor + 1}} = \\ &(4m + 3n - 9) - 2^{\lfloor log_2 m \rfloor + 1}\Gamma_{2^{\lfloor log_2 m \rfloor + 1}}. \end{aligned} \quad (17)$$

The term $\Gamma_{2^{\lfloor log_2 m \rfloor + 1}}$ in (17) can be computed using the definition previously given. According to that definition, it can be observed that $\Gamma_{2^{\lfloor log_2 m \rfloor + 1}}$ is the sum of eight *floor* functions of eight quotients with the same denominator $2^{\lfloor log_2 m \rfloor + 1}$ and where the numerators are integers smaller than $m$. However, the value of $2^{\lfloor log_2 m \rfloor + 1}$ is always greater than $m$, so the quotients are less than unity and all the *floor* functions are always zero. Therefore, the term $\Gamma_{2^{\lfloor log_2 m \rfloor + 1}} = 0$ and (17) will be $\mu_0 + 2\mu_1 + 2^2\mu_2 + \ldots + 2^{\lfloor log_2 m \rfloor}\mu_{\lfloor log_2 m \rfloor} = (4m + 3n - 9)$. Using this result and applying it to (13), it follows (18), that matches (6).

$$M_{\lfloor log_2 m \rfloor} = \left\lceil \frac{4m + 3n - 9}{2^{\lfloor log_2 m \rfloor}} \right\rceil \quad (18)$$

### B. Upper Bound for $T_X$ Delay

The delay of the multiplier given in (7) is the following:

$$T_A + \left( \lfloor log_2 m \rfloor + \left\lceil log_2 \left\lceil \frac{4m + 3n - 9}{2^{\lfloor log_2 m \rfloor}} \right\rceil \right\rceil \right) T_X \quad (19)$$

For type II irreducible pentanomials, $n \leq \lfloor m/2 \rfloor - 1$, so for *even* $m$ we have that $n \leq (m/2) - 1$ while that for *odd* $m$ we have $n \leq (m-1)/2 - 1$. The following operations can be done:

- *Even* $m$. We have $4m + 3n - 9 \leq 4m + 3((m/2) - 1) - 9 = 4m + (3m/2) - 12 = (11m/2) - 12 < (11m/2)$. Substituting this expression in the quotient in $M_{\lfloor log_2 m \rfloor}$ and using the fact that $2^{\lfloor log_2 m \rfloor + 1} > m$, then we have $((4m + 3n - 9)/2^{\lfloor log_2 m \rfloor}) < (11m/(2^{\lfloor log_2 m \rfloor} + 1)) < 11\not{m}/\not{m} = 11$ and therefore $\lceil (4m + 3n - 9)/2^{\lfloor log_2 m \rfloor} \rceil \leq 11$. Finally we will have that

$$\left\lceil log_2 \left\lceil \frac{4m + 3n - 9}{2^{\lfloor log_2 m \rfloor}} \right\rceil \right\rceil \leq \lceil log_2 \lceil 11 \rceil \rceil = 4. \quad (20)$$

- *Odd* $m$. We have $4m + 3n - 9 \leq 4m + 3(((m-1)/2) - 1) - 9 = ((11m - 9)/2) - 9 < (11m - 27)/2 < 11m/2$ and then we get the same results as in the previous case for *even* $m$.

Using the result given in (20), then we have $\lfloor log_2 m \rfloor + \lceil log_2 \lceil (4m + 3n - 9)/2^{\lfloor log_2 m \rfloor} \rceil \rceil \leq \lfloor log_2 m \rfloor + 4$ and using the property $1 + \lfloor log_2 m \rfloor = \lceil log_2 (m+1) \rceil$ we have finally

that the XOR delay of the multiplier can be upper bounded as follows, matching (8):

$$\lfloor log_2 m \rfloor + \left\lceil log_2 \left\lceil \frac{4m+3n-9}{2^{\lfloor log_2 m \rfloor}} \right\rceil \right\rceil \leq \lceil log_2(m+1) \rceil + 3 \tag{21}$$

## Appendix B
## Area Complexity

The XOR gates of the multiplier will be ① + ② + ③ − ④. These quantities are determined as follows:

① The functions $\mathbf{S_i}$ and $\mathbf{T_i}$ as given in (1), (2) are implemented as *binary trees* of 2-input XOR gates with a lower level of 2-input AND gates (corresponding to the $a_i b_j$ products). The number of AND and XOR gates for $\mathbf{S_i}$ are $i$ and $i-1$, while that for $\mathbf{T_i}$ they are $m-i-1$ and $m-i-2$, respectively [9]. The total contribution of $\mathbf{S_i}$ and $\mathbf{T_i}$ to the *space* complexity is $m^2$ AND and $(m-1)^2$ XOR gates [9]. Therefore, the total number of AND gates of the multiplier is $m^2$ and the number ① of XOR gates given by $\mathbf{S_i}$ and $\mathbf{T_i}$ in (1), (2) is $(m-1)^2$.

② The coefficients in Table I have been divided into seven sections (from Ⓐ to Ⓖ). In Section II, the number of $\mathbf{S_i}$ and $\mathbf{T_i}$ terms in the sums for each section was given. Taking into account these numbers, then the XOR gates in the product coefficients are as follows [9]: $4(n-1)$ for section Ⓐ, $14(=3+6+5)$ in Ⓑ, $7(n-3)$ in Ⓒ, 12 and 10 in Ⓓ and Ⓔ respectively; $4(m-2n-4)$ in Ⓕ and finally 3 in Ⓖ. Then the number ② of XOR gates needed for the sum of the $\mathbf{S_i}$ and $\mathbf{T_i}$ terms in the product coefficients is $4m+3n-2$. It must be noted that this number corresponds with the general case in which all the sections from Ⓐ to Ⓖ appear in Table I. There are some special cases for which the above complexity can be reduced. However, the number of such cases is negligible and they are not considered.

③ In order to compute the number ③ of XOR gates, we must first determine the number $p_i$ of times each $\mathbf{T_i}$ appears in Table I. It can be found that for the general case in which all the sections from Ⓐ to Ⓖ exist, there are $z-2$ terms $(\mathbf{T_0}, \ldots, \mathbf{T_{z-3}})$ that appear 4 times, $n$ terms $(\mathbf{T_{z-2}}$ and $\mathbf{T_z}, \ldots, \mathbf{T_{m-2}})$ that appear 7 times and the term $\mathbf{T_{z-1}}$ appearing 6 times. As previously stated, *one* occurrence of the $\mathbf{T_i}$ terms is already included in ①, so we must compute the XOR gates due to $\mathbf{T_0}, \ldots, \mathbf{T_{z-3}}$ appearing 3 times, $\mathbf{T_{z-2}}$ and $\mathbf{T_z}, \ldots, \mathbf{T_{m-2}}$ appearing 6 times and $\mathbf{T_{z-1}}$ appearing 5 times. If we define $\Phi_{0\ldots m-2} = \sum_{i=0}^{m-2} \Theta_i$ and $\Phi_{z-2\ldots m-2} = \sum_{i=z-2}^{m-2} \Theta_i$, where $\Theta_i$ is the number of XORs needed for the sum of the $\mathbf{T_i^j}$ terms for $\mathbf{T_i}$, then we can write that the number ③ of XOR gates is given by $\sum_{i=0}^{m-2}(p_i-1)\cdot\Theta_i = 3\cdot\Phi_{0\ldots m-2} + 3\cdot\Phi_{z-2\ldots m-2} - \Theta_{z-1}$. Using the equivalence (only in relation to the number of terms) $\mathbf{T_i} \equiv \mathbf{S_{m-1-i}}$ and denoting $\Upsilon_h = \sum_{i=1}^{h} \Psi_i$ where $\Psi_i$ is the number of XORs needed for the sum of the $\mathbf{S_i^j}$ terms for $\mathbf{S_i}$, then we can compute the number ③ of XOR gates using $\sum_{i=0}^{m-2}(p_i-1)\cdot\Theta_i = 3\cdot\Upsilon_{m-1} + 3\cdot\Upsilon_{n+1} - \Psi_n$. The number of XOR gates $\Psi_i$ for the sum of the $\mathbf{S_i^j}$ terms for $\mathbf{S_i}$ can be computed using the number of 1's in the binary configuration of $i$. For example, $\mathbf{S_{13}}$ in Table II is given in the form $\mathbf{S_{13}} = \mathbf{S_{13}^0} + \mathbf{S_{13}^2} + \mathbf{S_{13}^3}$ and therefore 2 XOR gates are needed to perform the additions of the $\mathbf{S_{13}^j}$ terms. The binary configuration of the subindex 13 in this case is (1101), i.e., with three 1's. Therefore the number of XOR gates $\Psi_{13}$ will be the number of 1's in the binary configuration of 13 minus 1. Using (3) and using the definition of *mod* operator ($x \bmod y = x - y\lfloor x/y \rfloor$), the *Hamming Weight* of $i$, can be computed as $H_i = \sum_{j=0}^{\lfloor log_2 i \rfloor} \lfloor i/2^j \rfloor \bmod 2 = \sum_{j=0}^{\lfloor log_2 i \rfloor} (\lfloor i/2^j \rfloor - 2 \cdot \lfloor i/2^{j+1} \rfloor) = i - (\lfloor i/2 \rfloor + \lfloor i/2^2 \rfloor + \ldots + \lfloor i/2^{\lfloor log_2 i \rfloor} \rfloor)$. Therefore the number of XOR gates $\Psi_i$ needed for the sum of the $\mathbf{S_i^j}$ terms for $\mathbf{S_i}$ will $H_i$ minus 1:

$$\Psi_i = \left( i - \sum_{j=1}^{\lfloor log_2 i \rfloor} \left\lfloor \frac{i}{2^j} \right\rfloor \right) - 1 = H_i - 1, \tag{22}$$

that matches (9). Using (22), then $\Upsilon_h$ can be computed:

$$\Upsilon_h = \sum_{i=1}^{h} \Psi_i = \frac{h(h-1)}{2} - \sum_{i=1}^{h} \sum_{j=1}^{\lfloor log_2 i \rfloor} \left\lfloor \frac{i}{2^j} \right\rfloor \tag{23}$$

that matches (10).

④ The number of XOR gates given by the shared groups $(\mathbf{T_i}, \mathbf{T_j})$ that appear in the product coefficients can be computed in a similar way as done in Section III-A. It can be found that for *even* $n$, the group $(\mathbf{T_{z-2}}, \mathbf{T_{z-1}})$ appears in three coefficients ($c_0$, $c_n$ and $c_{m-1}$) while that for *odd* $n$, the group $(\mathbf{T_{z-1}}, \mathbf{T_z})$ appears in the same coefficients. On the other hand, for *even* $m$, the groups $(\mathbf{T_0}, \mathbf{T_1}), (\mathbf{T_2}, \mathbf{T_3}), \ldots, (\mathbf{T_{m-4}}, \mathbf{T_{m-3}})$ appear in two coefficients while that for *odd* $m$, the groups $(\mathbf{T_1}, \mathbf{T_2}), (\mathbf{T_3}, \mathbf{T_4}), \ldots, (\mathbf{T_{m-4}}, \mathbf{T_{m-3}})$ also appear in two coefficients, in both cases excluding the previous groups for $n$ *even* or *odd*. This means that only one of each of the above groups must be implemented and therefore the other occurrences of the groups must not be taken into account. It must be noted that from the above $(\mathbf{T_i}, \mathbf{T_{i+1}})$ groups, the term with highest subindex gives the number of XOR gates to be shared. For example, using Table II it can be observed that for $\mathbf{T_0} = \mathbf{T_0^0} + \mathbf{T_0^2} + \mathbf{T_0^3}$ (with three terms) and $\mathbf{T_1} = \mathbf{T_1^2} + \mathbf{T_1^3}$ (two terms), the group $(\mathbf{T_0}, \mathbf{T_1})$ involves the sum of two terms $\mathbf{T_0^2} + \mathbf{T_1^2}$ and $\mathbf{T_0^3} + \mathbf{T_1^3}$ and therefore it requires 2 XOR gates. In order to compute the number of XORs, we must use the equivalence (only in relation to the number of terms) $\mathbf{T_i} \equiv \mathbf{S_{m-1-i}}$. Then the previous group $(\mathbf{T_{z-2}}, \mathbf{T_{z-1}})$ for *even* $n$ corresponds with $(\mathbf{S_{n+1}}, \mathbf{S_n})$, the group $(\mathbf{T_{z-1}}, \mathbf{T_z})$ for *odd* $n$ corresponds with $(\mathbf{S_n}, \mathbf{S_{n-1}})$, the groups $(\mathbf{T_0}, \mathbf{T_1}), \ldots, (\mathbf{T_{m-4}}, \mathbf{T_{m-3}})$ for *even* $m$ correspond with $(\mathbf{S_{m-1}}, \mathbf{S_{m-2}}), \ldots, (\mathbf{S_3}, \mathbf{S_2})$ and the groups $(\mathbf{T_1}, \mathbf{T_2}), \ldots, (\mathbf{T_{m-4}}, \mathbf{T_{m-3}})$ for *odd* $m$ correspond with $(\mathbf{S_{m-2}}, \mathbf{S_{m-3}}), \ldots, (\mathbf{S_3}, \mathbf{S_2})$. Furthermore, using the equivalence we have that the term with lowest subindex gives the number of XOR gates to be shared. Then the number ④ of XOR gates represented by the above shared groups is given by the number of 1's (Hamming Weight) in the binary configuration of $2, 4, 6, \ldots, m-2$ for *even* $m$ or of $2, 4, 6, \ldots, m-3$ for *odd* $m$ plus the Hamming Weight of $n$ for *even* $n$ or of $n-1$ for *odd* $n$. Therefore the number of XOR gates ④ given by the shared groups $(\mathbf{T_i}, \mathbf{T_j})$ that appear in the product coefficients is computed by

$$\sum_{i=2,4,6,\ldots}^{\frac{\iota}{\kappa}} H_i + H_{n^\dagger/(n-1)^\ddagger} = \Sigma_m + H_{\dagger/\ddagger} \qquad (24)$$

that matches (11). In (24), $\iota$ represents the limit $(m-2)$ of the summatory for *even* $m$, $\kappa$ represents the limit $(m-3)$ for *odd* $m$, $^\dagger$ represents the Hamming Weight of $n$ to be computed for *even* $n$ and $^\ddagger$ the Hamming Weight of $(n-1)$ for *odd* $n$.

## REFERENCES


[1] H. A. Curtis, *A New Approach to the Design of Switching Circuits*. Princeton, NJ, USA: Van Nostrand, 1962.

[2] J. Lin, J. Sha, Z. Wang, and L. Li, "Efficient decoder design for non-binary quasicyclic LDPC codes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 5, pp. 1071–1082, May 2010.

[3] T.-C. Chen, S.-W. Wei, and H.-J. Tsai, "Arithmetic unit for finite field $GF(2^m)$," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 3, pp. 828–837, Apr. 2008.

[4] J. L. Imaña, "Low latency $GF(2^m)$ polynomial basis multiplier," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, no. 5, pp. 935–946, May 2011.

[5] M. A. Hasan and M. Ebtedaei, "Efficient architectures for computations over variable dimensional galois fields," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 45, no. 11, pp. 1205–1211, Nov. 1998.

[6] J. L. Imaña, R. Hermida, and F. Tirado, "Low complexity bit-parallel polynomial basis multipliers over binary fields for special irreducible pentanomials," *Integration*, vol. 46, pp. 197–210, 2013.

[7] P. K. Meher, "Systolic and super-systolic multipliers for finite field $GF(2^m)$ based on irreducible trinomials," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 4, pp. 1031–1040, May 2008.

[8] C. L. Wang and J. L. Lin, "Systolic array implementation of multipliers for finite fields $GF(2^m)$," *IEEE Trans. Circuits Syst.*, vol. 38, no. 7, pp. 796–800, Jul. 1991.

[9] J. L. Imaña, "Efficient polynomial basis multipliers for Type II irreducible pentanomials," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 59, no. 11, pp. 795–799, Nov. 2012.

[10] R. Azarderakhsh, D. Jao, and H. Lee, "Common subexpression algorithms for space-complexity reduction of gaussian normal basis multiplication," *IEEE Trans. Inf. Theory*, vol. 61, no. 5, pp. 2357–2369, May 2015.

[11] S. T. J. Fenn, M. Benaissa, and D. Taylor, "$GF(2^m)$ multiplication and division over the dual basis," *IEEE Trans. Comput.*, vol. 45, no. 3, pp. 319–327, Mar. 1996.

[12] A. Reyhani-Masoleh and M. A. Hasan, "Low complexity bit parallel architectures for polynomial basis multiplication over $GF(2^m)$," *IEEE Trans. Comput.*, vol. 53, no. 8, pp. 945–959, Aug. 2004.

[13] T. Zhang and K. K. Parhi, "Systematic design of original and modified mastrovito multipliers for general irreducible polynomials," *IEEE Trans. Comput.*, vol. 50, no. 7, pp. 734–749, Jul. 2001.

[14] H. Fan and Y. Dai, "Fast bit parallel $GF(2^m)$ multiplier for all trinomials," *IEEE Trans. Comput.*, vol. 54, no. 4, pp. 485–490, Apr. 2005.

[15] E. D. Mastrovito, "VLSI architectures for multiplication over finite fields $GF(2^m)$," in *Proc. 6th Int'l Conf. Appl. Algebra, Algebraic Algorithms, Error-Correcting Codes (AAECC-6)*, New York, Jul. 1988, pp. 297–309, Rome, Italy: Springer-Verlag.

[16] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. Hoboken, NJ, USA: Wiley, 1999.

[17] A. Halbutogullari and Ç. K. Koç, "Mastrovito multiplier for general irreducible polynomials," *IEEE Trans. Comput.*, vol. 49, no. 5, pp. 503–518, May 2000.

[18] F. Rodríguez-Henríquez and Ç. K. Koç, "Parallel multipliers based on special irreducible pentanomials," *IEEE Trans. Comput.*, vol. 52, no. 12, pp. 1535–1542, Dec. 2003.

[19] V. B. Afanasyev, "Complexity of VLSI implementation of finite field arithmetic," in *Proc. II Int. Workshop Algebraic Combinatorial Coding Theory*, Sep. 1990, pp. 6–7.

[20] J. L. Imaña, R. Hermida, and F. Tirado, "Low complexity bit-parallel multipliers based on a class of irreducible pentanomials," *IEEE Trans. Very Large Scale Integr. (VLSI) Systems*, vol. 14, no. 12, pp. 1388–1393, Dec. 2006.

[21] S.-M. Park, K.-Y. Chang, D. Hong, and C. Seo, "New efficient bit-parallel polynomial basis multiplier for special pentanomials," *Integration*, vol. 47, pp. 130–139, 2014.


**José L. Imaña** received the M.Sc. and Ph.D. degrees in physics from Complutense University, Madrid, Spain, in 1989 and 2003, respectively. He was an Electronic Design Engineer at the Madrid Institute of Technology, Spain. He is currently with the Department of Computer Architecture and Systems Engineering at Complutense University, where he was promoted to an Associate Professor with tenure in 2006. He has been the promoter and cofounder of the International Workshop on the Arithmetic of Finite Fields (WAIFI). His research interests include algorithms and VLSI architectures for computations in finite fields, cryptography, computer arithmetic, and reconfigurable computing.